

李春葆 蒋　林 陈良臣
喻丹丹 曾　平　编著

电子商务网站开发教程
——基于C#+ASP.NET

清华大学出版社

21世纪高等学校电子商务专业规划教材

电子商务网站开发教程
——基于C＃＋ASP .NET

李春葆　蒋　林　陈良臣　喻丹丹　曾　平　编著

清华大学出版社
北　京

内 容 简 介

本书以 Visual Studio 2012 和 SQL Server 2012 为环境介绍电子商务网站的开发方法，内容包括电子商务概述、电子商务网站开发环境配置、ASP .NET 网站结构、HTML 与 CSS、JavaScript 编程基础、C# 编程基础、ASP .NET 控件、ASP .NET 内置对象、主题设计、母版页设计、导航设计、ASP .NET 数据库编程和 ASP .NET 数据控件，最后介绍一个简单电子商务网站 OnRetS 的开发过程和相关技术。

本书内容翔实，循序渐进，并提供了全面而丰富的教学资源，可作为各类高等院校计算机及相关专业"电子商务网站开发"和"ASP .NET 动态网站开发"课程的教学用书，也可供计算机应用人员和计算机爱好者参考。

图书在版编目(CIP)数据

电子商务网站开发教程：基于 C#＋ASP .NET/李春葆等编著. —北京：清华大学出版社，2016
21 世纪高等学校电子商务专业规划教材
ISBN 978-7-302-43198-5

Ⅰ. ①电…　Ⅱ. ①李…　Ⅲ. ①电子商务－网站－开发－教材　Ⅳ. ①F713.36 ②TP393.092

中国版本图书馆 CIP 数据核字(2016)第 034758 号

责任编辑：魏江江　赵晓宁
封面设计：
责任校对：焦丽丽
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：31.25　　**字　　数**：764 千字
版　　次：2016 年 4 月第 1 版　　**印　　次**：2016 年 4 月第1 次印刷
印　　数：1～　000
定　　价：　.00 元

产品编号：067893-01

出 版 说 明

电子商务是以信息网络技术为手段，以商品交换为中心的商务活动，是“互联网＋”的杰作之一。特别是在2015年初的政府工作报告中，李克强总理首次提出“制订‘互联网＋’行动计划”，大大推进电子商务在我国的蓬勃发展，改造和影响着众多传统行业。电子商务系统是保证以电子商务为基础的网上交易实现体系。

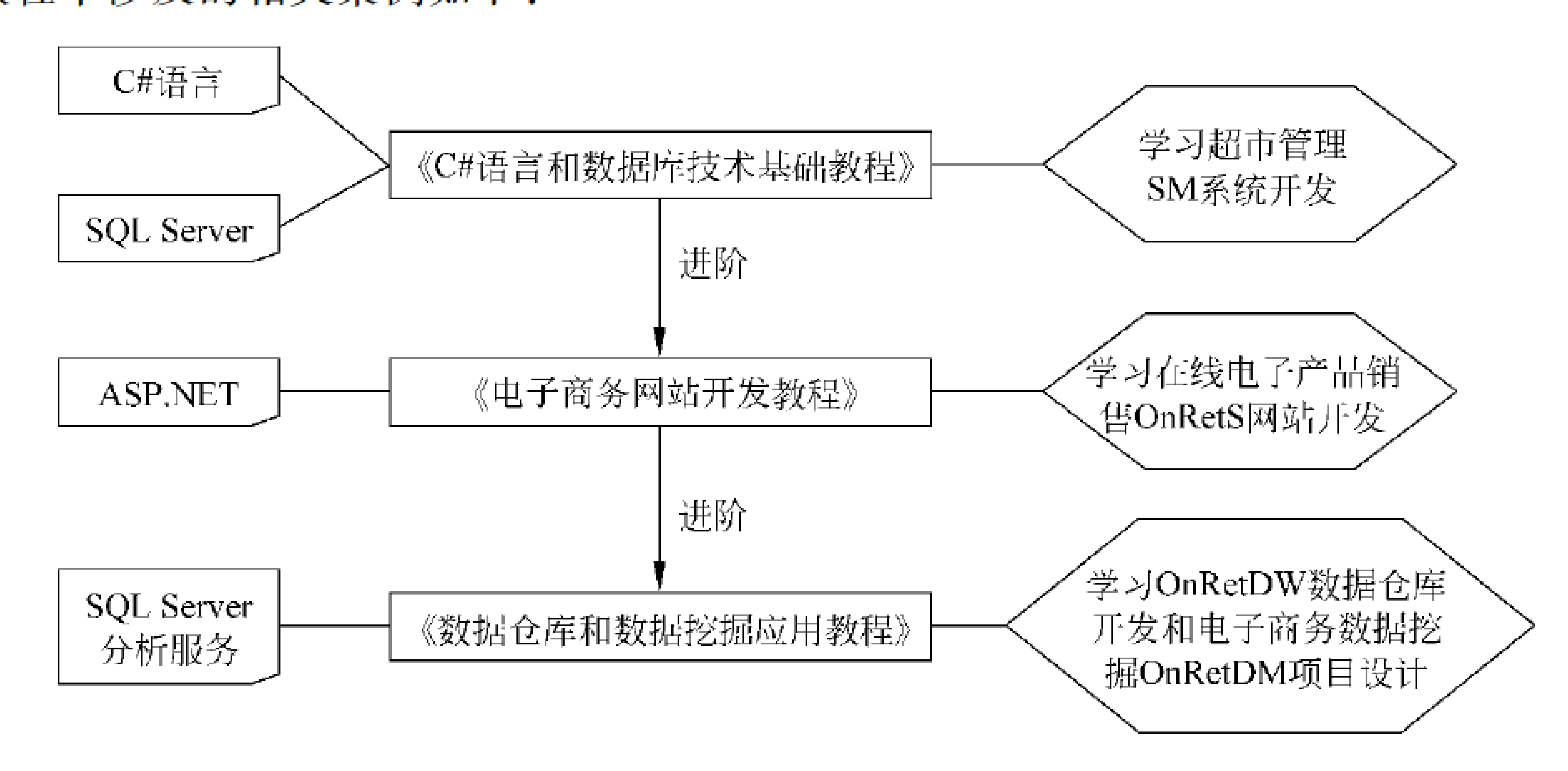
电子商务应用的快速发展，需要大量的专业技术人员，据专家测算，未来10年我国电子商务人才缺口达到200万。为了加快电子商务系统人才培养，我们以国家卓越工程师计划为契机规划并出版了本系列教材。

电子商务网站是电子商务系统的核心，电子商务网站开发涉及多方面的技术。本系列教材以Visual Studio为开发环境、以案例为向导，全面介绍电子商务网站的开发技术，涵盖的教程如下：

- C＃语言和数据库技术基础教程；
- 电子商务网站开发教程；
- 数据仓库和数据挖掘应用教程。

本系列教材具有专业培养定位清晰、可操作性强的特点。《C＃语言和数据库技术基础》从零基础开始，循序渐进学习C＃语言的基本语法、面向对象编程、Windows窗体应用程序设计、SQL Server数据库操作、C＃访问数据库方法以及Windows界面的电子商务系统开发技术。《电子商务网站开发教程》以ASP .NET为背景，学习动态网站的开发技术。《数据仓库和数据挖掘应用教程》学习数据仓库数据和电子商务数据分析技术。

教程中涉及的相关案例如下：



本系列教材是武汉大学计算机学院和解放军理工大学在探索电子商务人才培养并结合国家卓越工程师计划的教学实践中总结和提炼的教学成果。教学改革是教育工作永恒不变的主题，也是需要不断探索的课题，需要不断地努力实践和完善。本系列教材虽然经过细致的编写与校订，仍然难免有疏漏和不足之处，需要不断地补充、修订和完善，我们热情欢迎使用本系列教材的教师、学生和读者朋友提出宝贵意见和建议，使之更臻成熟。

前　　言

ASP .NET 是微软公司提出的动态网站开发技术，具有易学易用、开发效率高等特点，是目前主流的 Web 开发环境之一。

本书内容如下：

第 1 章为电子商务概述，介绍电子商务的相关概念和技术。

第 2 章为电子商务网站开发环境配置，介绍 ASP .NET 开发电子商务网站的环境和配置过程。

第 3 章为 ASP .NET 网站结构，介绍 ASP .NET 网站和网页的基本结构。

第 4 章为 HTML 和 CSS，介绍设计静态网页所需要的 HTML 和 CSS 基本知识。

第 5 章为 JavaScript 编程基础，介绍采用 JavaScript 编写客户端应用程序的方法。

第 6 章为 C# 编程基础，介绍采用 C# 编写服务器端应用程序的方法。

第 7 章为 ASP .NET 控件，介绍 ASP .NET 提供的服务器控件及其使用方法。

第 8 章为 ASP .NET 内置对象，介绍 ASP .NET 内置对象在网页设计中的应用。

第 9 章为主题、母版页和导航设计，介绍设计一致性网页的基本技术。

第 10 章为 ASP .NET 数据库编程，介绍 ADO .NET 访问 SQL Server 数据库的基本方法。

第 11 章为 ASP .NET 数据控件，介绍 ASP .NET 提供的数据源控件和数据绑定控件的使用方法。

第 12 章为电子商务网站开发实例，介绍一个简单的电子商务网站 OnRetS 的开发过程和相关技术。

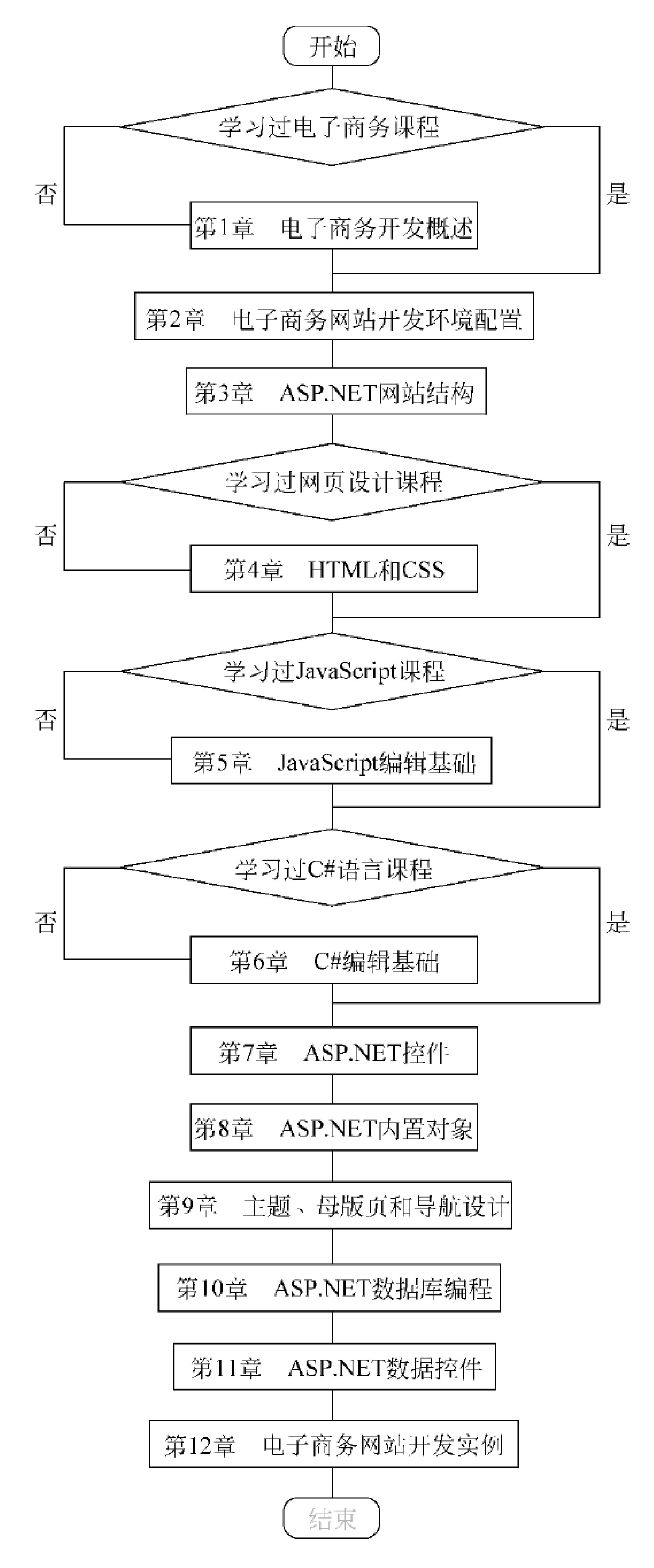
书中提供了大量的练习题和上机实验题供读者选用，附录 A 给出了部分练习题参考答案，附录 B 给出了所有上机实验题参考答案。

本书紧扣 ASP .NET 网站开发所需要的知识、技能和素质要求，以技术应用能力培养为主线构建教材内容，具有以下特色：

- 内容全面、丰富：在内容讲授上力求翔实和全面，细致解析每个知识点和各知识点的联系。
- 条理清晰、讲解透彻：从介绍 ASP .NET 的基本概念出发，由简单到复杂，循序渐进介绍网站的开发过程。
- 精选实例、实用性强：列举了大量的应用示例，读者通过上机模仿可以提高使用 ASP .NET 网站开发能力。
- 力求从入门到精通：本书起点只需要读者学习过一门高级语言程序设计课程，通过实训能够达到如 OnRetS 中小型电子商务网站的开发水平。
- 配套教学资源丰富：提供教学 PPT、书中所有【练一练】实例源代码、相关数据库文件和电子商务网站 OnRetS 的源程序。书中实例网站均采用文件系统方式创建，便于读

者打开和调试。配套的教学资源可以从清华大学出版社网站下载。

本书推荐的学习流程如下：



本书在编写过程中得到武汉大学教务部教改项目的资助，解放军理工大学和清华大学出版社给予了大力支持，连续多届选课的同学提出了许多宝贵的建议，编者在此表示衷心感谢。

编　者

2015 年 10 月

目　　录

第 1 章　电子商务开发概述 …… 1

1.1　什么是电子商务 …… 1

1.1.1　电子商务的定义 …… 2

1.1.2　电子商务系统和电子商务网站 …… 2

1.1.3　电子商务网站的功能、特点和分类 …… 3

1.2　电子商务网站的技术基础 …… 5

1.2.1　计算机网络及其类型 …… 5

1.2.2　WWW、互联网和因特网 …… 5

1.2.3　Web 的系统结构和工作原理 …… 7

1.2.4　Web 网页、网页文件和网站 …… 9

1.2.5　静态网页和动态网页 …… 9

1.2.6　Web 网页开发技术 …… 11

1.3　电子商务网站的开发过程 …… 12

1.3.1　电子商务网站开发步骤 …… 12

1.3.2　网站规划与设计 …… 13

1.3.3　网站建设 …… 14

1.3.4　网站发布和网站的管理与维护 …… 15

1.4　练习题 …… 16

第 2 章　电子商务网站开发环境配置 …… 18

2.1　电子商务网站开发环境 …… 18

2.2　安装和配置 Visual Studio 2012 …… 19

2.2.1　安装 Visual Studio 2012 …… 19

2.2.2　设置 Visual C# 开发环境 …… 23

2.3　安装和使用 SQL Server 2012 …… 25

2.3.1　下载并安装 SQL Server 2012 …… 25

2.3.2　SQL Server 2012 的主要管理工具 …… 37

2.3.3　创建数据库 school …… 37

2.3.4　在 school 数据库中创建 3 个数据表 …… 39

2.4　练习题 …… 41

第 3 章 ASP .NET 网站结构 …… 43
3.1 ASP .NET 网站 …… 43
3.1.1 ASP .NET 网站的构成 …… 44
3.1.2 ASP .NET 网站创建过程 …… 45
3.2 ASP .NET 网页 …… 46
3.2.1 设计 ASP .NET 网页的过程 …… 46
3.2.2 ASP .NET 网页的执行过程 …… 50
3.2.3 ASP .NET 网页代码编写模型 …… 51
3.2.4 ASP .NET 网页的基本结构 …… 56
3.3 ASP .NET 网站配置文件 …… 58
3.3.1 web.config 文件 …… 58
3.3.2 system.web 配置节 …… 59
3.3.3 自定义配置节 …… 61
3.4 练习题 …… 62
第 4 章 HTML 和 CSS …… 65
4.1 HTML 文档结构 …… 65
4.1.1 HTML 文档的基本结构 …… 65
4.1.2 HTML 文档的头部标记 …… 66
4.1.3 HTML 文档的主体标记 …… 67
4.2 HTML 文档主体中的常用标记 …… 69
4.2.1 HTML 基础标记 …… 69
4.2.2 HTML 格式标记 …… 70
4.2.3 HTML 表格标记 …… 72
4.2.4 HTML 样式/节标记 …… 73
4.2.5 HTML 列表标记 …… 75
4.2.6 HTML 超链接标记 …… 77
4.2.7 HTML 图像标记 …… 79
4.2.8 HTML 框架标记 …… 81
4.2.9 HTML 表单标记 …… 84
4.3 CSS 设计 …… 89
4.3.1 CSS 是什么 …… 89
4.3.2 CSS 样式设计 …… 89
4.3.3 CSS 样式的组织方式 …… 93
4.3.4 使用 Visual Studio 样式生成器设计样式 …… 95
4.3.5 CSS 方框模型 …… 97
4.3.6 网页页面布局 …… 99
4.4 练习题 …… 101
4.5 上机实验题 …… 104
第 5 章 JavaScript 编程基础 …… 105
5.1 JavaScript 概述 …… 105

5.1.1　JavaScript 的特点 …… 105

5.1.2　HTML 文档中引入 JavaScript 脚本代码 …… 106

5.2　JavaScript 的数据类型和运算符 …… 108

5.2.1　变量的定义方式 …… 109

5.2.2　JavaScript 的基本数据类型 …… 109

5.2.3　JavaScript 的其他数据类型 …… 111

5.2.4　值变量和引用变量 …… 116

5.2.5　JavaScript 的运算符 …… 117

5.3　JavaScript 的控制语句 …… 118

5.3.1　选择条件语句 …… 119

5.3.2　循环控制语句 …… 122

5.4　JavaScript 的函数设计 …… 124

5.4.1　创建和调用 JavaScript 函数 …… 124

5.4.2　函数的参数 …… 125

5.4.3　函数的返回值 …… 126

5.4.4　变量 …… 126

5.4.5　异常处理语句 …… 127

5.5　事件编程 …… 129

5.5.1　JavaScript 事件和事件处理方法 …… 129

5.5.2　事件处理方法的调用方法 …… 130

5.6　浏览器对象 …… 132

5.6.1　浏览器对象模型 …… 133

5.6.2　window 对象 …… 133

5.6.3　document 对象 …… 136

5.6.4　history 对象 …… 142

5.6.5　location 对象 …… 144

5.6.6　navigator 对象 …… 145

5.6.7　screen 对象 …… 146

5.7　练习题 …… 148

5.8　上机实验题 …… 152

第 6 章　C# 编程基础 …… 153

6.1　C# 概述 …… 153

6.1.1　C# 的特点 …… 154

6.1.2　C# 和 ASP .NET …… 154

6.1.3　HTML 文档和 C# 脚本代码 …… 154

6.2　C# 的数据类型 …… 155

6.2.1　值类型 …… 155

6.2.2　引用类型 …… 156

6.2.3　变量定义 …… 157

6.3　C# 的运算符 …… 158

6.3.1 C♯常用的运算符 …… 158
6.3.2 运算符的优先级 …… 159
6.3.3 装箱和拆箱运算 …… 160
6.4 结构体类型和枚举类型 …… 160
6.4.1 结构体类型 …… 161
6.4.2 枚举类型 …… 162
6.5 C♯的控制语句 …… 163
6.5.1 选择控制语句 …… 163
6.5.2 循环控制语句 …… 165
6.6 C♯的数组和集合 …… 167
6.6.1 一维数组 …… 167
6.6.2 二维数组 …… 168
6.6.3 集合 …… 170
6.7 异常处理和命名空间 …… 170
6.7.1 异常处理 …… 170
6.7.2 命名空间 …… 172
6.8 面向对象程序设计 …… 173
6.8.1 设计类 …… 174
6.8.2 创建类对象 …… 176
6.8.3 构造函数和析构函数 …… 179
6.8.4 静态成员 …… 181
6.8.5 属性设计 …… 182
6.8.6 方法设计 …… 183
6.8.7 委托和事件 …… 186
6.9 C♯中常用类和结构体 …… 186
6.9.1 String 类 …… 186
6.9.2 Math 类 …… 187
6.9.3 Convert 类 …… 188
6.9.4 数据类型转换 …… 188
6.9.5 DateTime 结构体 …… 189
6.10 继承和接口 …… 190
6.10.1 继承设计 …… 190
6.10.2 接口设计 …… 194
6.11 练习题 …… 196
6.12 上机实验题 …… 199
第 7 章 ASP .NET 控件 …… 200
7.1 ASP .NET 控件概述 …… 200
7.1.1 什么是 ASP .NET 控件 …… 200
7.1.2 HTML 控件和 HTML 服务器控件 …… 201
7.1.3 Web 服务器控件 …… 202

7.2 Web 标准服务器控件 …… 202
7.2.1 Web 标准服务器控件的分类 …… 203
7.2.2 Web 标准控件的公共属性、公共方法和公共事件 …… 203
7.3 常用的表单控件 …… 204
7.3.1 Label 控件 …… 205
7.3.2 TextBox 控件 …… 205
7.3.3 Button 控件 …… 206
7.3.4 Image 控件 …… 207
7.3.5 HyperLink 控件 …… 207
7.3.6 CheckBox 控件 …… 208
7.3.7 RadioButton 控件 …… 208
7.4 常用的列表控件 …… 212
7.4.1 DropDownList 控件 …… 212
7.4.2 ListBox 控件 …… 213
7.4.3 CheckBoxList 控件 …… 214
7.4.4 RadioButtonList 控件 …… 215
7.5 常用的其他标准控件和组件 …… 219
7.5.1 FileUpload 控件 …… 219
7.5.2 发送邮件组件 …… 221
7.6 ASP .NET 验证控件 …… 224
7.6.1 验证控件概述 …… 225
7.6.2 RequiredFieldValidator 控件 …… 226
7.6.3 CompareValidator 控件 …… 226
7.6.4 RangeValidator 控件 …… 227
7.6.5 RegularExpressionValidator 控件 …… 227
7.6.6 CustomValidator 控件 …… 228
7.6.7 ValidationSummary 控件 …… 234
7.7 练习题 …… 234
7.8 上机实验题 …… 237
第 8 章 ASP .NET 内置对象 …… 238
8.1 ASP .NET 对象概述 …… 238
8.1.1 常用的 ASP .NET 对象 …… 238
8.1.2 网页生命周期 …… 239
8.2 Page 对象 …… 241
8.2.1 Page 对象的属性、方法和事件 …… 241
8.2.2 Page 对象的应用 …… 242
8.3 Response 对象 …… 245
8.3.1 Response 对象的属性和方法 …… 246
8.3.2 Response 对象的应用 …… 247
8.4 Request 对象 …… 248

8.4.1 Request 对象的属性和方法 …… 248
8.4.2 Request 对象的应用 …… 250
8.5 Server 对象 …… 253
8.5.1 Server 对象的属性和方法 …… 253
8.5.2 Server 对象的应用 …… 254
8.6 Application 对象 …… 256
8.6.1 Application 对象的属性、方法和事件 …… 256
8.6.2 Global.asax 文件 …… 258
8.6.3 Application 对象的应用 …… 259
8.7 Session 对象 …… 261
8.7.1 Session 对象的属性、方法和事件 …… 261
8.7.2 Session 对象的应用 …… 263
8.8 Cookie 对象 …… 263
8.8.1 Cookie 对象的属性 …… 264
8.8.2 Cookie 对象的应用 …… 264
8.9 ViewState 对象 …… 266
8.9.1 ViewState 对象概述 …… 267
8.9.2 ViewState 对象的应用 …… 268
8.10 练习题 …… 270
8.11 上机实验题 …… 272
第 9 章 主题、母版页和导航设计 …… 273
9.1 设计和应用主题 …… 273
9.1.1 主题概述 …… 273
9.1.2 创建主题 …… 274
9.1.3 应用主题 …… 276
9.1.4 禁用主题 …… 278
9.2 设计和应用母版页 …… 279
9.2.1 母版页和内容页概述 …… 279
9.2.2 创建母版页 …… 280
9.2.3 创建内容页 …… 282
9.3 站点导航设计 …… 286
9.3.1 站点导航概述 …… 287
9.3.2 创建站点地图 …… 287
9.3.3 站点导航控件概述 …… 288
9.3.4 TreeView 控件 …… 289
9.3.5 TreeView 控件的应用 …… 292
9.4 练习题 …… 297
9.5 上机实验题 …… 299
第 10 章 ASP .NET 数据库编程 …… 300
10.1 数据库概述 …… 300

10.1.1　关系数据库的基本概念 …… 300
10.1.2　结构化查询语言 …… 301
10.2　ADO .NET 模型 …… 305
10.2.1　ADO .NET 模型简介 …… 305
10.2.2　ADO .NET 数据库的访问流程 …… 308
10.3　ADO .NET 的数据访问对象 …… 308
10.3.1　SqlConnection 对象 …… 309
10.3.2　SqlCommand 对象 …… 311
10.3.3　SqlDataReader 对象 …… 317
10.3.4　SqlDataAdapter 对象 …… 320
10.4　DataSet 对象 …… 322
10.4.1　DataSet 对象概述 …… 323
10.4.2　DataTable 对象 …… 323
10.4.3　DataSet 对象的应用 …… 324
10.5　练习题 …… 325
10.6　上机实验题 …… 328
第 11 章　ASP .NET 数据控件 …… 329
11.1　数据控件概述 …… 329
11.1.1　数据控件的用途 …… 330
11.1.2　ASP .NET 有哪些数据控件 …… 330
11.2　SqlDataSource 控件 …… 331
11.2.1　SqlDataSource 控件概述 …… 331
11.2.2　SqlDataSource 控件的应用 …… 334
11.2.3　SQL 注入攻击 …… 342
11.3　列表绑定控件 …… 343
11.3.1　列表绑定控件概述 …… 343
11.3.2　列表绑定控件的应用 …… 344
11.4　GridView 控件 …… 345
11.4.1　GridView 控件概述 …… 346
11.4.2　GridView 控件的基本设计 …… 347
11.4.3　GridView 控件的复杂设计 …… 353
11.5　DetailsView 控件 …… 372
11.5.1　DetailsView 控件概述 …… 373
11.5.2　DetailsView 控件的应用 …… 374
11.6　ObjectDataSource 控件 …… 376
11.6.1　ObjectDataSource 控件概述 …… 376
11.6.2　ObjectDataSource 控件的应用 …… 377
11.7　练习题 …… 387
11.8　上机实验题 …… 389

第12章 电子商务网站开发实例 ………… 390
12.1 OnRetS网站的需求分析 ………… 390
12.1.1 OnRetS网站的功能 ………… 390
12.1.2 OnRetS网站的主要业务流程 ………… 392
12.2 OnRetS网站结构设计 ………… 393
12.3 数据库设计 ………… 394
12.3.1 创建数据库OnRet ………… 394
12.3.2 创建数据表 ………… 395
12.4 网站公共模块设计 ………… 397
12.4.1 web.config配置文件 ………… 397
12.4.2 CommDB.cs类文件 ………… 398
12.4.3 StyleSheet.css样式文件 ………… 400
12.4.4 MasterPage.master母版页 ………… 401
12.4.5 公共网页dispinfo.aspx ………… 401
12.5 主页设计 ………… 402
12.6 游客功能网页设计 ………… 404
12.6.1 游客功能主页设计 ………… 404
12.6.2 “用户注册”功能网页设计 ………… 406
12.6.3 “查看(浏览)商品”功能网页设计 ………… 410
12.7 顾客功能网页设计 ………… 415
12.7.1 顾客功能主页设计 ………… 415
12.7.2 “选购商品放入购物车”功能网页设计 ………… 417
12.7.3 “编辑我的购物车”功能网页设计 ………… 423
12.7.4 “购物车结算”功能网页设计 ………… 424
12.7.5 “查看我的订单”功能网页设计 ………… 427
12.7.6 “撤销尚未处理的订单”功能网页设计 ………… 429
12.7.7 “订单商品评价”功能网页设计 ………… 430
12.7.8 “更改我的信息”功能网页设计 ………… 433
12.7.9 “更改我的密码”功能网页设计 ………… 434
12.8 管理员功能网页设计 ………… 434
12.8.1 管理员功能主页设计 ………… 434
12.8.2 “添加新用户信息”功能网页设计 ………… 435
12.8.3 “编辑用户信息”功能网页设计 ………… 435
12.8.4 “查看顾客信息”功能网页设计 ………… 437
12.8.5 “临时封杀顾客信息”功能网页设计 ………… 438
12.8.6 “查看顾客订单信息”功能网页设计 ………… 440
12.8.7 “商品库存预警”功能网页设计 ………… 442
12.8.8 “商品库存报警”功能网页设计 ………… 444
12.8.9 “商品下架”功能网页设计 ………… 446
12.8.10 “按商品分类统计”功能网页设计 ………… 447

12.8.11　“按商品子类统计”功能网页设计…………………………………… 448
12.8.12　“按商品品牌统计”功能网页设计…………………………………… 449
12.8.13　“设置顾客学历数据”功能网页设计………………………………… 450
12.8.14　“设置顾客地区数据”功能网页设计………………………………… 452
12.8.15　“设置商品类型数据”功能网页设计………………………………… 453
12.8.16　“删除下架的商品信息”功能网页设计……………………………… 453
12.8.17　“系统初始化”功能网页设计………………………………………… 455
12.9　操作员功能网页设计 …………………………………………………… 456
12.9.1　操作员功能主页设计 ………………………………………………… 456
12.9.2　“添加新型号商品信息”功能主页设计 ……………………………… 457
12.9.3　“更新老商品信息”功能主页设计 …………………………………… 459
12.9.4　“查看新订单”功能主页设计 ………………………………………… 460
12.9.5　“新订单处理”功能主页设计 ………………………………………… 463
12.9.6　“新订单结算处理”功能主页设计 …………………………………… 465
附录 A　部分练习题参考答案………………………………………………… 467
附录 B　上机实验题参考答案………………………………………………… 475
参考文献………………………………………………………………………… 484

第 1 章　电子商务开发概述

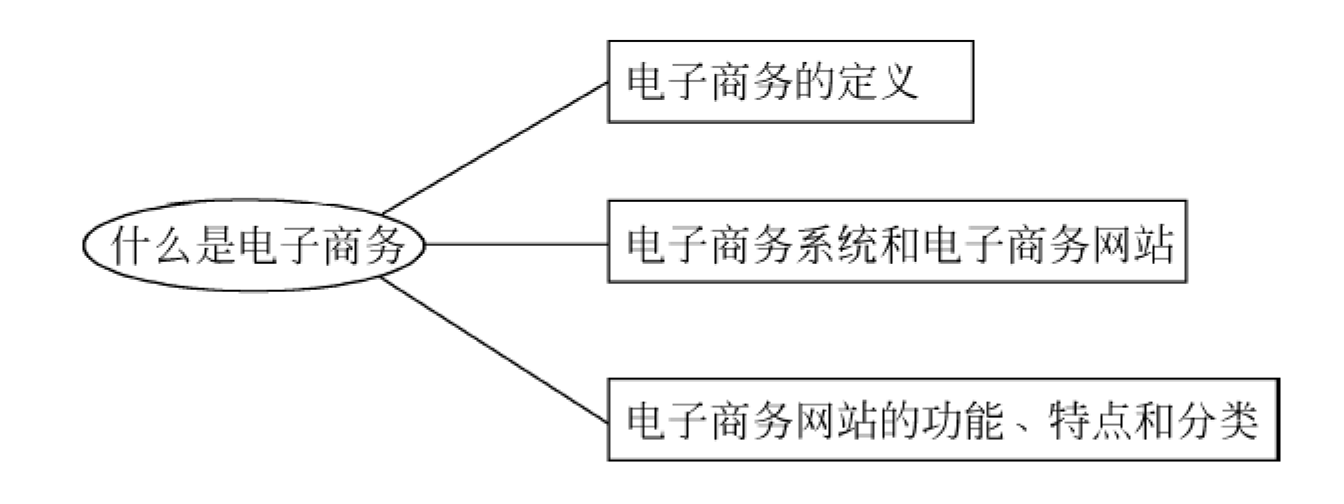


本章指南

- 什么是电子商务
- 电子商务网站的技术基础
- 电子商务网站的开发过程

1.1　什么是电子商务

知识梳理



1.1.1 电子商务的定义

各国政府、学者、企业界人士根据自己所处的地位和对电子商务参与的角度和程度的不同，给出了许多不同的定义。

联合国经济合作与发展组织（OECD）对电子商务的定义是："电子商务是发生在开放网络上的包含企业之间（Business to Business）、企业和消费者之间（Business to Consumer）的商业交易。"

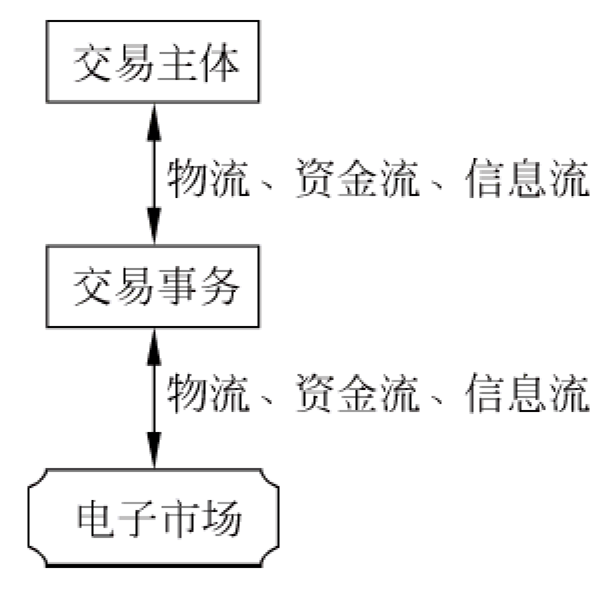


图 1.1 电子商务的概念模型

《中国电子商务蓝皮书》（2001 年）认为：电子商务指通过 Internet 完成的商务交易，交易的内容可分为商品交易和服务交易。交易是指货币和商品的易位，交易要有信息流、资金流和物流的支持，如图 1.1 所示。

电子商务通常是指在全球各地广泛的商业贸易活动中，在开放的网络环境下，基于浏览器/服务器应用方式，买卖双方不谋面地进行各种商贸活动，实现消费者的网上购物、商户之间的网上交易和在线电子支付以及各种商务活动、交易活动、金融活动和相关的综合服务活动的一种新型的商业运营模式。

总之，电子商务是以信息网络技术为手段，以商品交换为中心的商务活动；也可理解为在互联网（Internet）、企业内部网（Intranet）和增值网（Value Added Network，VAN）上以电子交易方式进行交易活动和相关服务的活动，是传统商业活动各环节的电子化、网络化和信息化。电子商务的本质是商务，技术只是电子商务的手段。

电子商务的基本功能概括为 3C，即内容管理（Content Management）、协同处理（Collaboration）与交易服务（Commerce）三大类。

电子商务的基本特征如下：

- 运用现代网络信息技术进行商务管理活动；
- 电子商务活动的参与者相对于传统商务有很大的变化；
- 电子商务既贯穿商品交易的全过程，也贯穿整个社会经济活动的始终；
- 电子商务范围的全球化；
- 电子商务是直接性的经济活动；
- 电子商务是个性化的经济活动。

1.1.2 电子商务系统和电子商务网站

电子商务系统是指在 Internet 和其他网络的基础上，以实现企业电子商务活动为目标，满足企业生产、销售、服务等生产和管理的需要，支持企业的对外业务协作，从运作、管理和决策等层次全面提高企业信息化水平，为企业提供商业智能的计算机系统。图 1.2 展示了金融电子商务系统的框架。

电子商务网站是指通过网站建设技术发布、展示商品信息，实现电子交易，并通过网络开展与商务活动有关的各种售前和售后服务，全面实现电子商务功能的网站。电子商务网站的手段是"发布、展示商品信息"。

电子商务系统是基于 Internet 并支持企业价值链增值的信息系统，而网站仅仅是这一系统的一个部分或技术手段之一。电子商务系统作为一个整体，不仅包括企业开展商务活动的

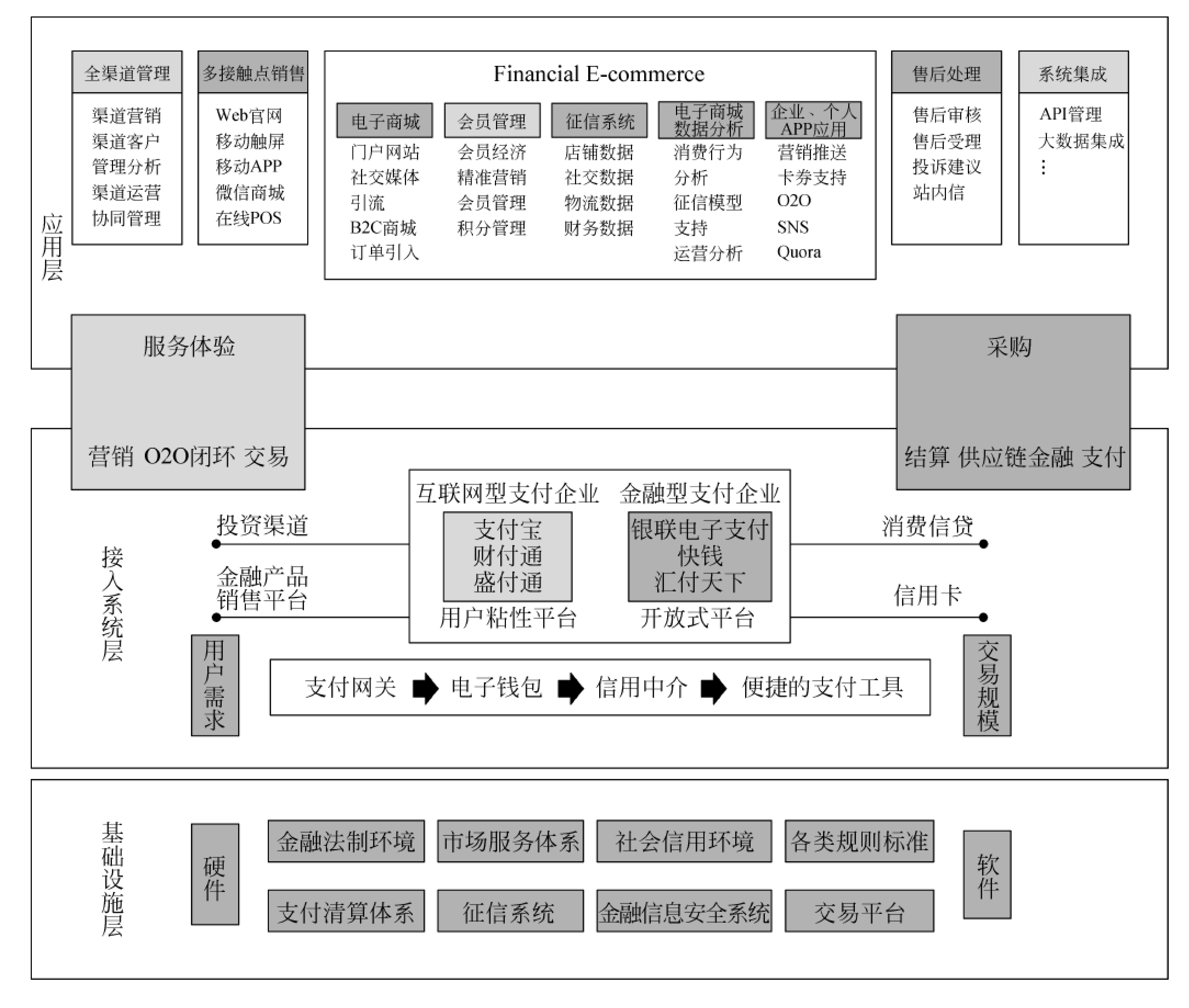


图 1.2　金融电子商务系统框架示例

外部电子化环境，而且包括企业内部商务活动的电子化环境，这两部分必须结合起来才能满足企业在 Internet 上开展商务活动的需要。

因此，可以将网站视为企业电子商务系统的一个重要组成部分。需要说明的是，企业的电子商务系统因企业的规模、服务方式不同而使其功能差异很大，但大多数的电子商务系统都是利用网站与客户进行交互的。

另外，一些企业电子商务系统的规模较小且商务处理功能很弱(如仅仅实现企业形象宣传功能)，因此这些电子商务系统从外部就表现为电子商务网站的形式。

1.1.3　电子商务网站的功能、特点和分类

1. 功能

电子商务网站的基本功能如下：

- 企业形象宣传；
- 信息编辑；
- 咨询洽谈；
- 网上商品订购；
- 网上支付；

- 用户信息管理；
- 服务传递；
- 销售业务信息管理。

2. 特点

电子商务网站的特点如下。

- 商务性：就是解决电子商务是做什么的这一核心问题，也可以说电子商务的实质是商务活动。
- 服务性：电子商务的服务性要围绕方便客户这一原则。
- 集成性：新技术和新概念的集成、处理商务活动时的整体性和统一性。
- 安全性：安全性是电子商务生存的基础条件
- 可扩展性：是指其系统具备扩展能力，能够满足电子商务活动繁忙时系统速度、流量压力承受。
- 协调性：商务活动是一种协调过程，需要整体的协调，电子商务系统才能正常有效运转。

3. 分类

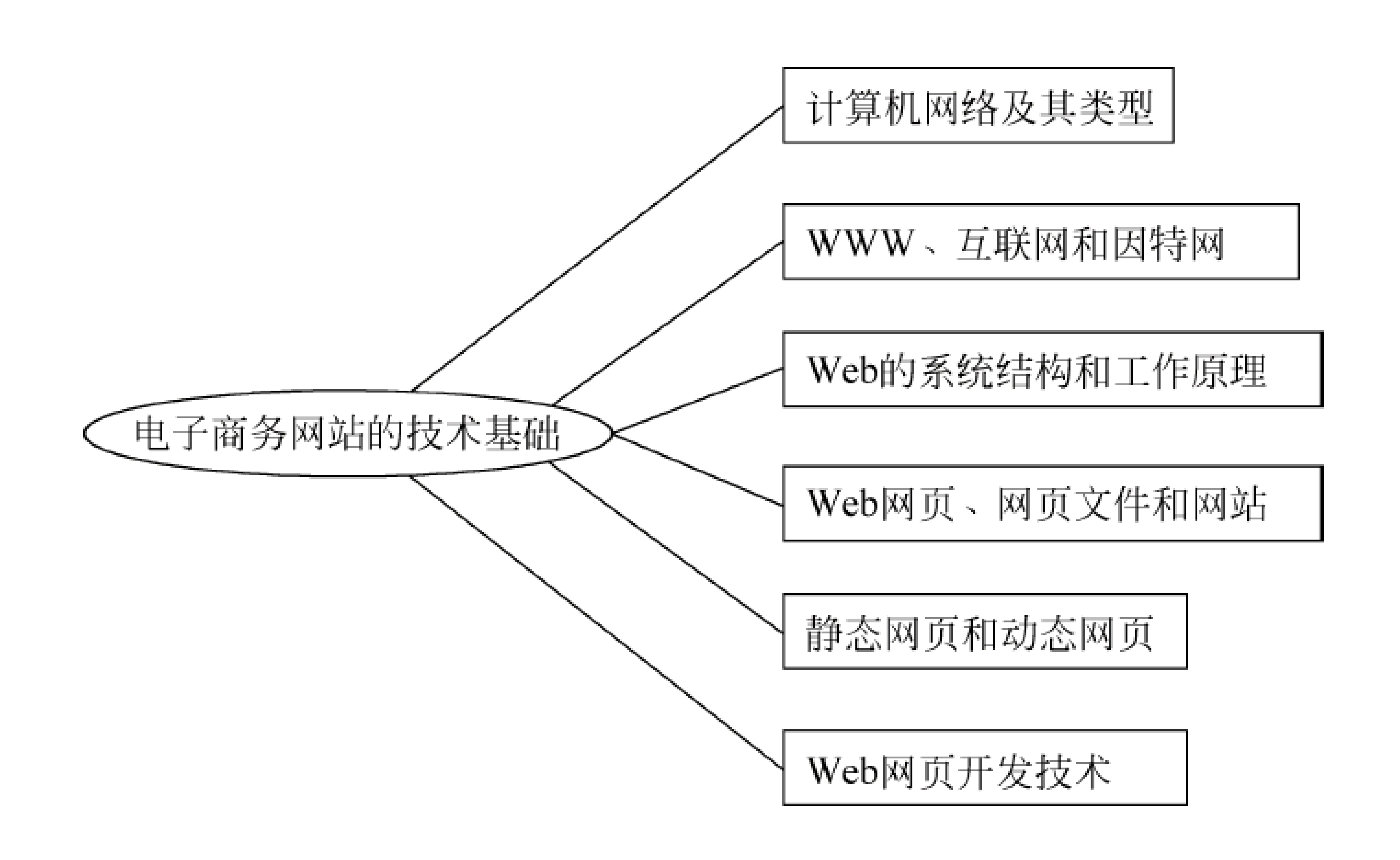
按参加电子商务活动的交易主体将电子商务网站分类如下：

- 企业与消费者(BtoC或B2C)。B2C即企业与消费者之间的电子商务。这是消费者利用互联网直接参与经济活动的形式，类似于商业电子化的零售商务。随着互联网的出现，网上销售迅速地发展起来，B2C就是企业通过网络销售产品或服务给个人消费者。
- 企业与企业(BtoB或B2B)。B2B电子商务是指以企业为主体，在企业之间进行的电子商务活动。B2B电子商务是电子商务的主流，也是企业面临激烈的市场竞争、改善竞争条件、建立竞争优势的主要方法。B2B主要是针对企业内部以及企业与上下游协力厂商之间的资讯整合，并在互联网上进行的企业与企业间交易。借由企业内部网建构资讯流通的基础，及外部网络结合产业的上中下游厂商，达到供应链(SCM)的整合。
- 消费者与消费者(CtoC或C2C)。C2C也称“私对私”是一种典型的电子商务类型，它是指个人对个人的交易形式。
- B2B2C(Business to Business to Customer)。第一个B指交易平台，即提供卖方与买方的交易平台，同时提供优质的附加服务，第二个B指广义的卖方(即成品、半成品、材料提供商等)，C即指消费者。B2B2C平台绝非简单的中介，而是提供高附加值服务的渠道机构，拥有客户管理、信息反馈、数据库管理、决策支持等功能的服务平台。B2B2C定义包括现存的B2C和C2C平台的商业模式，更加综合化。
- 其他类型，如企业与政府(BtoG或B2G)、消费者与政府(CtoG或C2G)、B2M(Business to Marketing)即面向市场营销的电子商务企业、O2O(Online to Offline)即线上线下电子商务、消费者对企业(C2B)和企业对团队(B2T)等。

另外，按电子商务活动的运作方式可以将电子商务网站分为直接交易型电子商务网站和间接交易型电子商务网站。

1.2　电子商务网站的技术基础

知识梳理



1.2.1　计算机网络及其类型

计算机网络是指将地理位置不同的具有独立功能的多台计算机及其外部设备，通过通信线路连接起来，在网络操作系统，网络管理软件及网络通信协议的管理和协调下，实现资源共享和信息传递的计算机系统。

虽然计算机网络类型的划分标准各种各样，但是从地理范围划分是一种公认的通用网络划分标准。按这种标准可以把各种计算机网络划分为以下类型：

- 局域网(Local Area Network，LAN)是指在某一区域内由多台计算机互联成的计算机组。一般是在方圆几千米以内。局域网可以实现文件管理、应用软件共享、打印机共享、工作组内的日程安排、电子邮件和传真通信服务等功能。局域网是封闭型的，可以由办公室内的两台计算机组成，也可以由一个公司内的上千台计算机组成。
- 城域网(Metropolitan Area Network，MAN)是在一个城市范围内所建立的计算机通信网，属宽带局域网。它的一个重要用途是用作骨干网，通过它将位于同一城市内不同地点的主机、数据库，以及 LAN 等互相连接起来。
- 广域网(Wide Area Network，WAN)也称远程网。通常跨接很大的物理范围，所覆盖的范围从几十千米到几千千米，它能连接多个城市或国家，或横跨几个洲并能提供远距离通信，形成国际性的远程网络。

要想让两台计算机进行通信，必须使它们采用相同的信息交换规则。把在计算机网络中用于规定信息的格式以及如何发送和接收信息的一套规则称为网络协议或通信协议，如 TCP/IP 协议。

迄今为止，计算机网络经过了 4 个阶段的发展，即远程终端联机阶段、计算机网络阶段、计算机网络互联阶段和国际互联网与信息高速公路阶段。

1.2.2　WWW、互联网和因特网

WWW 是环球信息网(World Wide Web)的缩写(亦为 Web、W3 等)，后面主要使用 Web

名称,中文名字为万维网、环球网等。它起源于 1989 年 3 月,由欧洲粒子物理实验室研究发展起来的主从结构分布式超媒体系统,最初开发设计的目的是为该实验室的物理学家们提供一种共享和信息的工具。

Web 的特点如下:

- Web 是图形化和易于导航:Web 可以提供将图形、音频、视频信息集合于一体的特性。另外,Web 使用一种超文本(hypertext)链接技术。超文本可以是 Web 网页上的任意的一个元素,由它指向因特网上的其他 Web 元素,所以,Web 是易于导航的,浏览用户就可以在各网页各站点之间进行方便地浏览。
- Web 与平台无关:无论系统平台是什么,都可以通过因特网访问 Web。浏览 Web 对系统平台没有什么限制。无论从 Windows 平台或 UNIX 平台等都可以访问 Web。对 Web 的访问是通过一种叫做浏览器的软件实现的,如 Netscape 的 Navigator、Microsoft 的 Explorer 等。
- Web 是分布式的:大量的图形、音频和视频信息会占用相当大的磁盘空间,甚至无法预知信息的多少。对于 Web 没有必要把所有信息都放在一起,信息可以放在不同的站点上。只需要在浏览器中指明这个站点就可以了。使在物理上并不一定在一个站点的信息在逻辑上一体化,从用户来看这些信息是一体的。
- Web 是动态的:由于各 Web 站点包含站点本身的信息,信息的提供者可以经常对站点上的信息进行更新,所以 Web 站点上的信息是动态的。

要理解 Web,必须了解互联网(internet)和因特网(Internet)这两个十分容易混淆的名字。《现代汉语词典》(2002 年增补本)将互联网定义为"指由若干计算机网络相互连接而成的网络",将因特网定义为"全球最大的一个电子计算机互联网,是由美国的 ARPA 发展演变而来的",因特网的英文首字母用大写表示。

也就是说,互联网是网络与网络之间所串连成的庞大网络,这些网络以一组通用的协议相连,形成逻辑上的单一巨大国际网络。因特网和其他类似的由计算机相互连接而成的大型网络系统,都可算是互联网,因特网只是互联网中最大的一个。

Web 是无数个网络站点和网页的集合,它们在一起构成了因特网最主要的部分。实际上 Web 是多媒体的集合,是由超链接连接而成的。人们通常通过网络浏览器上网观看的就是 Web 的内容。所以,Web 并不等同因特网,它只是因特网所能提供的服务其中之一,是靠着因特网运行的一项服务。或者说,因特网指的是一个硬件的网络,而 Web 更倾向于一种浏览网页的功能。

另外有一个名字是 Intranet,称为企业内部网,是因特网技术在企业内部的应用。它实际上是采用因特网技术建立的企业内部网络,在一个企业或组织的内部并为其成员提供信息的共享和交流等服务,如文件传输、电子邮件等。Intranet 在内部网络上采用 TCP/IP 作为通信协议,利用因特网的 Web 模型作为标准信息平台,同时建立防火墙把内部网和因特网分开。当然 Intranet 并非一定要和因特网连接在一起,它完全可以自成一体作为一个独立的网络。

万维网联盟(World Wide Web Consortium,W3C),又称 W3C 理事会。1994 年 10 月在麻省理工学院(MIT)计算机科学实验室成立。万维网联盟的创建者是万维网的发明者蒂姆·伯纳斯-李。

1.2.3 Web 的系统结构和工作原理

1. 客户机/服务器模式

Web 的系统结构采用的是客户机/服务器结构模式，如图 1.3 所示。Web 可以让客户机(常用浏览器)通过因特网访问服务器上的网页。在这个结构中，所有资源由一个全局“统一资源标识符”(URI)来标识。简单地说，URL 就是 Web 地址，俗称“网址”。其基本格式如下：

```
protocol :// hostname[:port] /path/
```

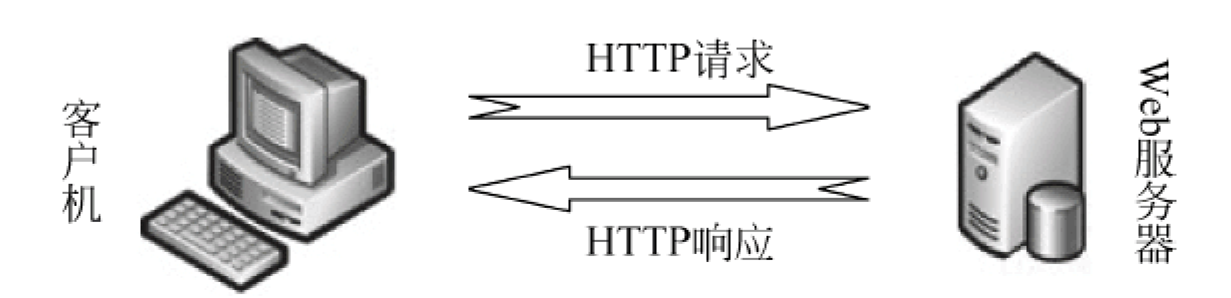
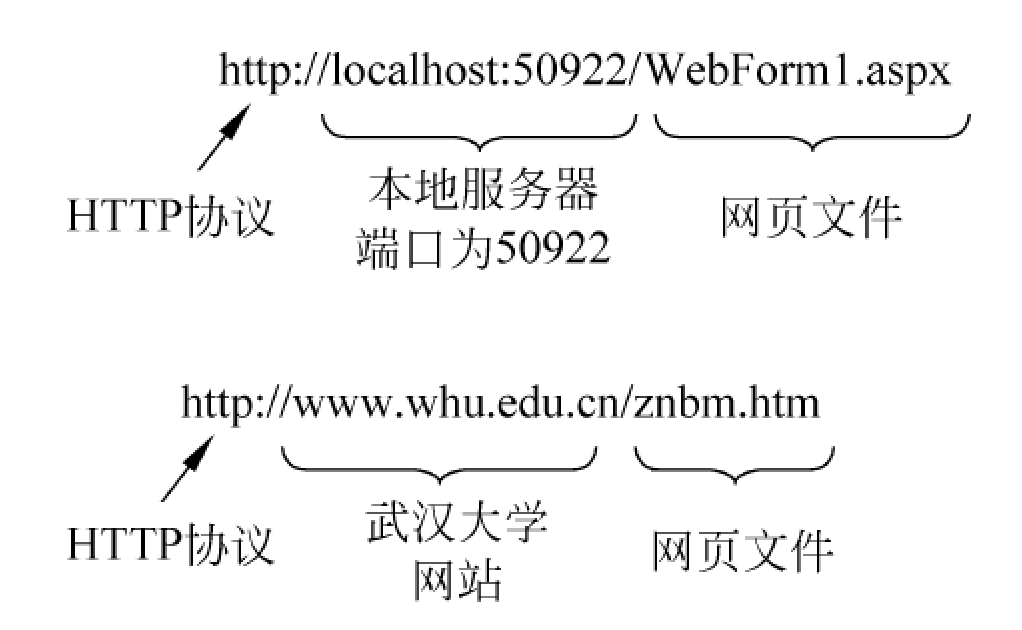


图 1.3 Web 的客户机/服务器结构模式

各部分说明如下。

- protocol：指定使用的文件传输协议，通常为 HTTP。
- hostname(主机名)：是指存放资源的服务器的域名系统(DNS)主机名或 IP 地址。
- port(端口号)：整数，可选，省略时使用方案的默认端口，各种传输协议都有默认的端口号，如 HTTP 的默认端口为 80。如果输入时省略，则使用默认端口号。
- path(路径)：由零或多个“/”符号隔开的字符串，一般用来表示主机上的一个目录或文件名(如果省略目录，服务器在网站主目录查找文件；如果省略文件名，服务器查找命名为 index.html、index.htm、default.html 或 default.htm 的文件)。

例如：



URL 提供了位于本地 Web 服务器或远程 Web 服务器上的信息，通过使用 HTTP 来管理这些信息，用户就可以在 Web 上创建对 URL 的引用了。

广义上讲，凡是提供服务的一方称为服务端，而接受服务的一方称为客户端。例如，当读者在浏览新浪网站主页时，新浪网站主页所在的服务器称为服务端，而读者自己的计算机就称为客户端。

只要在计算机上安装有接受服务的软件，这台计算机就变成一台服务器。例如，在一台计算机上安装有数据库服务器组件(如 SQL Server)，它就是一台数据库服务器。Web 服务器是指具有允许它们接受和响应来自客户端计算机的请求的特定软件的计算机，Web 服务器允许用户通过 Internet/Intranet 共享信息。

在一台计算机上安装有客户端软件，这台计算机就变成一台客户机。在因特网中客户端

软件主要有浏览器，如 IE 浏览器或百度浏览器等。

这里指的服务器和客户机并不是从硬件上划分的，如果一台计算机上既安装有服务器软件，又安装有客户机软件，则它既是服务器又是客户机，也就是说它既可以作为服务端又可以作为客户端。如果此时本机的客户端访问本机的服务端，相对该客户端而言，该服务端称为本机服务端。

2. HTTP 协议

HTTP 是一种以 TCP/IP 通信协议为基础的应用协议，它提供了在 Web 服务器和客户端浏览器之间传输信息的一种机制。也就是说，HTTP 负责规定客户端浏览器和服务器怎样互相交流的。HTTP 协议的主要特点可概括如下。

- 支持客户机/服务器模式。
- 简单快速：客户机向服务器请求服务时，只需传送请求方法和路径。由于 HTTP 协议简单，使得 HTTP 服务器的程序规模小，因而通信速度很快。
- 灵活：HTTP 允许传输任意类型的数据对象。
- 无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。
- 无状态：HTTP 协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另外，在服务器不需要先前信息时它的应答就较快。

Web 使用 HTTP 协议传输各种超文本网页和数据。HTTP 协议的会话过程包括如下 4 个步骤。

① 建立连接：客户端的浏览器向服务端发出建立连接的请求，服务端给出响应就可以建立连接了。

② 发送请求：客户端按照协议的要求通过连接向服务端发送自己的请求。客户端通常采用 HTTP URL 向服务器端发出访问请求。

③ 给出应答：服务端按照客户端的要求给出应答，把结果(HTML 文件)返回给客户端。也就是说，服务端将选中的 HTML 文档通过该连接传输到客户端，并将之在浏览器中显示出来。

④ 关闭连接：客户端接到应答后关闭连接。也就是说，HTML 文档传到客户端后，服务器将会立即自动地终止这个 TCP/IP 连接。

注意：*在向客户端发送所请求文件的同时，服务器并没有存储关于该客户的任何状态信息。即便某个客户端在几秒钟内再次请求同一个对象，服务器也不会说自己刚刚给它发送了这个对象。*

例如，某客户端发送请求的 URL 为 http://www. Website. com/somepath/index. html。

假设相应的 index. html 网页由 1 个基本 HTML 文件和 5 个 JPEG 图像文件构成，而且所有这些对象都存放在同一台服务器主机中。则完整的会话过程如下：

① 客户机初始化一个与服务器主机 www. website. com 中的服务器的 TCP 连接。服务器使用默认端口号 80 监听来自客户机的连接建立请求。

② 客户机经由与 TCP 连接相关联的本地套接字发出一个 HTTP 请求消息。这个消息中包含路径名/somepath/index. html。

③ 服务器经由与 TCP 连接相关联的本地套接字接收这个请求消息，再从服务器主机的内存或硬盘中取出对象/somepath/index. html，经由同一个套接字发出包含该对象的响应消息。

④ 服务器告知 TCP 关闭这个 TCP 连接(不过 TCP 要到客户机收到刚才这个响应消息

之后才会真正终止这个连接)。

⑤ 客户经机由同一个套接字接收这个响应消息。TCP 连接随后终止。该消息标明所封装的对象是一个 HTML 文件。客户机从中取出这个文件,加以分析后发现其中有 5 个 JPEG 对象的引用。

⑥ 给每一个引用到的 JPEG 对象重复步骤①～④。

1.2.4　Web 网页、网页文件和网站

Web 网页简称为网页或页面,是指因特网上按照 HTML(超文本标记语言)格式组织起来的文件,在通过因特网进行信息查询时以信息页面的形式出现,它包括图形、文字、声音和视像等信息。网页是网站的基本信息单位,是 Web 的基本文档。它由文字、图片、动画、声音等多种媒体信息组成,通过链接实现与其他网页或网站的关联和跳转。

网页文件是用 HTML 编写的,可在 Web 上传输,能被浏览器识别显示的文本文件,其扩展名是 htm 和 html 等。

网站通常由众多不同内容的网页构成,具有独立域名,网页的内容可体现网站的全部功能。通常把进入网站首先看到的网页称为首页或主页(homepage)。

1.2.5　静态网页和动态网页

在因特网中,最常见的就是 Web 网页。一般说来,出现在浏览器中的 Web 网页不外乎有两种,即静态网页和动态网页。

1. 静态网页

所谓静态网页就是指那些不能够接收用户输入信息的 Web 网页,其内容是静态的,唯一的响应就是接收鼠标点击超链接后显示所连接的网页。当用户用鼠标点击其中一个超链接后,就会在浏览器中显示所连接的网页信息。

静态网页采用 HTML 标记语言编写的,静态网页文件通常采用 html 或 htm 等扩展名。

例如,采用 Windows 中的记事本编写以下静态网页代码并存储在 spage.html 文件中(位于 D:\电子商务\CH1\目录下):

```
<html>
   <head>
       <title>一个静态网页</title>
   </head>
   <body>
      <center>
          <font face = "黑体"><h2>- 电子商务网站 -</h2></font>
          <font face = "宋体"><h3>欢迎光临</h3></font>
      </center>
   </body>
</html>
```

双击该文件,在 IE 浏览器(默认浏览器)中显示的结果如图 1.4 所示。在浏览器中右击,在出现的快捷菜单中选择“查看源”命令,显示的源代码与上述代码相同。

从中看到,静态网页中没有程序代码,只有 HTML 标记。

静态网页的工作过程如图 1.5 所示,其基本步骤如下:

① 用户通过客户机浏览器输入网址并回车来发出 Web 请求。

② 服务器收到静态网页请求。



图 1.4 一个静态网页在 IE 浏览器中显示的结果

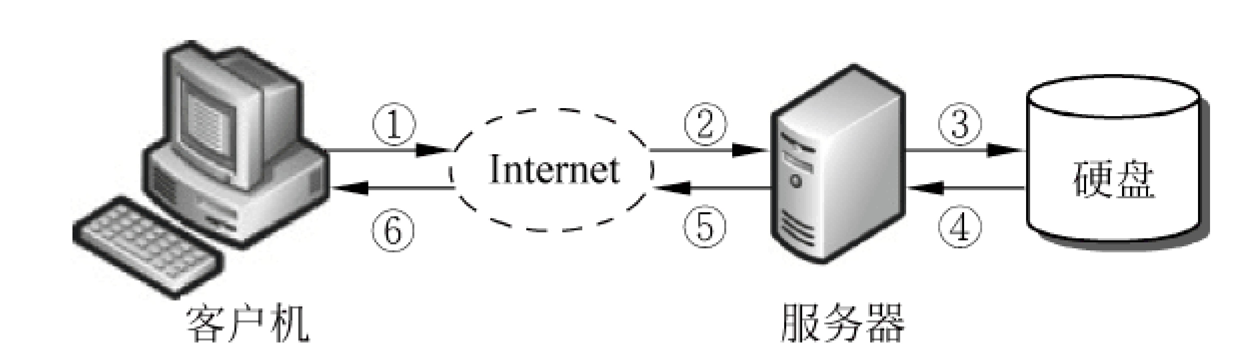


图 1.5 静态网页的工作过程

③ 服务器从硬盘的指定位置查找相应的 HTML 文件。

④ 将硬盘中找到的 HTML 文件返回给服务器。

⑤ 服务器向客户机返回该请求的文件。

⑥ 客户机浏览器收到请求的文件，并解析这些 HTML 代码将它显示出来。

从中看到，Web 服务器的主要功能是找到用户要访问的网页文件，不做任何改动直接传给客户端。也就是说，静态网页是实实在在保存在服务器上的文件，每个网页都是一个独立的文件。由于静态网页没有数据库的支持，内容相对稳定，因此容易被 Web 服务器查找，访问效率高。

需要指出的是，静态网页中可以包含客户端脚本，常见的客户端脚本语言有 JavaScript 或 VBScript 等。客户端脚本在一个特定的网页中改变界面以及行为或响应鼠标或键盘操作，如滚动字幕。在这种情况下，客户端会发生动态行为，但客户端生成的内容都在用户的本地计算机系统中，也就是说，这类网页的动态行为都在客户端进行的，是在客户端“动”而网页本身动态生成的，所以仍将这类网页归入静态网页。

2. 动态网页

所谓动态网页就是执行时用户可以输入所允许的各种信息，以实现人机交互，它能够根据不同的时间、不同的访问者而显示不同的内容。采用动态网页技术的网站可以实现更多的功能，如用户注册、用户登录、在线调查、用户管理、订单管理等。

动态网页中不仅含有 HTML 标记，还含有相关的程序代码。例如，采用 ASP .NET 设计的动态网页，网页的后缀名是 aspx。

动态网页的工作过程与图 1.5 类似，其步骤如下：

① 用户通过客户机浏览器输入网址并回车来发出 Web 请求。

② 服务器收到动态网页请求。

③ 服务器从硬盘的指定位置查找相应的动态网页文件

④ 将硬盘中找到的动态网页文件返回给服务器，服务器执行源代码，生成 HTML 文件。

⑤ 服务器向客户机返回该 HTML 文件。

⑥ 客户机浏览器收到请求的文件，并以图形方式将 HTML 标记显示在计算机屏幕上。

从中看到，对于动态网页，服务器的主要功能是通过文件系统找到用户要访问的动态网页文件，执行该网页文件的程序代码，产生 HTML 文件，再将该 HTML 文件传给客户机。

3. 静态网页和动态网页的比较

和静态网页的服务器相比，动态网页的服务器不仅要查找动态网页文件，还要解释执行其中的程序代码(对于 ASP .NET 网页，这种程序代码是由 ASP .NET 引擎执行的，对于静态网页，不需要这种引擎)，将含有程序代码的动态网页转化为标准的静态网页，然后将静态网页发送给客户机。也就是说，动态网页实际上并不是独立存在于服务器上的网页文件，只有当用户请求时服务器才返回一个完整的网页，这个返回的网页事先并不存在，而是由引擎(如 ASP

.NET 引擎)生成的。

归纳起来,静态网页和动态网页的比较如表 1.1 所示。从中看出,静态网页和动态网页各有优缺点,在实际应用中,开发人员根据任务的需要选择设计哪种类型的网页。

表 1.1　静态网页和动态网页的比较

比较项	静 态 网 页	动 态 网 页
内容	网页内容固定	网页内容动态生成
含程序代码	无	含 C# 或 VB 等程序代码
后缀	. htm、. html、. shtml 等	. asp、. jsp、. php、. cgi、. aspx 等
优点	无须系统实时生成,网页风格灵活多样	日常维护简单,更改结构方便,交互性能强
缺点	交互性能较差,日常维护繁琐	需要大量的系统资源合成网页
数据库	不支持	支持

1.2.6　Web 网页开发技术

WWW 是一种典型的分布式应用架构,其应用中的每一次信息交换都要涉及客户端和服务端两个层面。因此,Web 网页开发技术大体上也可以被分为 Web 客户端技术和 Web 服务器端技术两大类。

Web 客户端的主要任务是展现信息内容,而 HTML 语言则是信息展现的最有效载体之一。最初的 HTML 语言只能在浏览器中展现静态的文本或图像信息,满足不了人们对信息丰富和多样性的强烈需求,因此,由静态技术向动态技术的转变成为了 Web 客户端技术演进的必然趋势。目前,支持 Web 客户端动态技术主要有 VBScript 和 JavaScript 脚本语言。

与客户端技术从静态向动态的演进过程类似,Web 服务器端的网页开发技术也是由静态向动态逐渐发展、完善起来的。最早的 Web 服务器简单地响应浏览器发来的 HTTP 请求,并将存储在服务器上的 HTMIL 文件返回给浏览器。

第一种真正使服务器能根据执行时的具体情况,动态生成 HTML 页面的技术是 CGI (Common Gateway Interface,通用网关接口)技术。CGI 技术允许服务器端的应用程序根据客户端的请求,动态生成 HTML 网页,这使客户端和服务器端的动态信息交换成为了可能。

1994 年出现了 PHP(Hypertext Preprocessor,超文本预处理器)语言,它将 HTML 代码和 PHP 指令合成为完整的服务易端动态页面,Web 应用的开发者可以用一种更加简便、快捷的方式实现动态 Web 功能。

1996 年,微软公司借鉴 PHP 的思想,在其 Web 服务器 IIS 3. 0(Internet Information Server,Internet 信息服务器)中引入了 ASP(Active Server Pages,活跃服务器页面)技术。ASP 使用的脚本语言 VBScript 和 JavaScript,借助 Microsoft Visual Studio 等开发工具在市场上的成功,ASP 迅速成为了 Windows 系统下 Web 服务器端的主流开发技术。

1997 年,Servlet 技术问世。1998 年,JSP 技术诞生。Servlet 和 JSP 的组合(还可以加上 JavaBean 技术)让 Java 开发者同时拥有了类似 CGI 程序的集中处理功能和类似 PHP 的 HTML 嵌入功能。

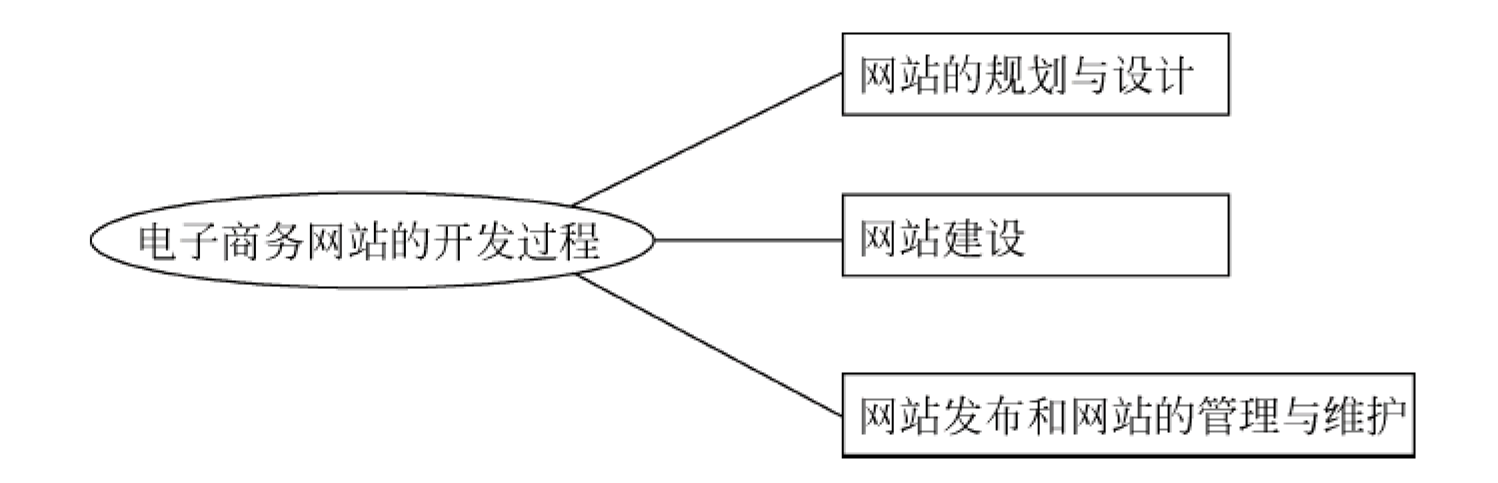
2000 年,微软公司推出了基于 .NET Framework 的 ASP .NET 1. 0 版本,2002 年推出了 ASP .NET 1. 1 版本,2005 年推出了 ASP .NET 2. 0 版本,2008 年推出了 ASP .NET 3. 5 版本,2010 年推出了 ASP .NET 4. 0 版本,2012 年又推出了 ASP .NET 4. 5 版本。

下面简要介绍常见的几种 Web 网页开发技术:

- CGI：一种早期的动态网页技术。可以使用不同的程序设计语言编写适合的 CGI 程序，如 VB、Delphi 或 C/C++ 等。虽然 CGI 技术已经发展成熟而且功能强大，但由于编程困难、效率低下、修改复杂，所以逐渐被新技术所取代。
- ASP：是微软公司开发的服务器端脚本环境，内置于 IIS 3.0 及以后版本之中，通过 ASP 可结合 HTML 网页、ASP 指令和 ActiveX 组件建立动态、交互且高效的 Web 服务器应用程序。有了 ASP，就不必担心客户浏览器是否能执行所编写的代码，因为所有程序都将在服务器端执行，包括所有嵌在普通 HTML 中的脚本程序。当程序执行完毕后，服务器仅将执行结果返回给客户浏览器，这样也减轻了客户浏览器的负担，大大提高了交互的速度。ASP 3.0 是经典 ASP 的最后一个版本。
- PHP：是一种易于学习和使用的服务器端脚本语言。只需要很少的编程知识就能使用 PHP 建立一个真正交互的 Web 网站。PHP 不需要特殊的开发环境，不仅支持多种数据库，还支持多种通信协议。
- JSP：与 ASP 技术非常相似，两者都提供在 HTML 代码中混合某种程序代码、由语言引擎解释执行程序代码的功能。与 ASP 一样，JSP 中的 Java 代码均在服务器端执行。
- ASP .NET：是继 ASP 后推出的全新动态网页制作技术，是建立在 .NET Framework 的公共语言运行库上的，可用于在服务器上生成功能强大的 Web 应用程序。它在性能上比 ASP 强很多，与 PHP 和 JSP 相比也存在明显的优势。

1.3　电子商务网站的开发过程

知识梳理



1.3.1　电子商务网站开发步骤

电子商务网站开发总的来说需要经历 4 个步骤，分别是网站的规划与设计、网站建设、网站发布和网站的管理与维护，如图 1.6 所示。

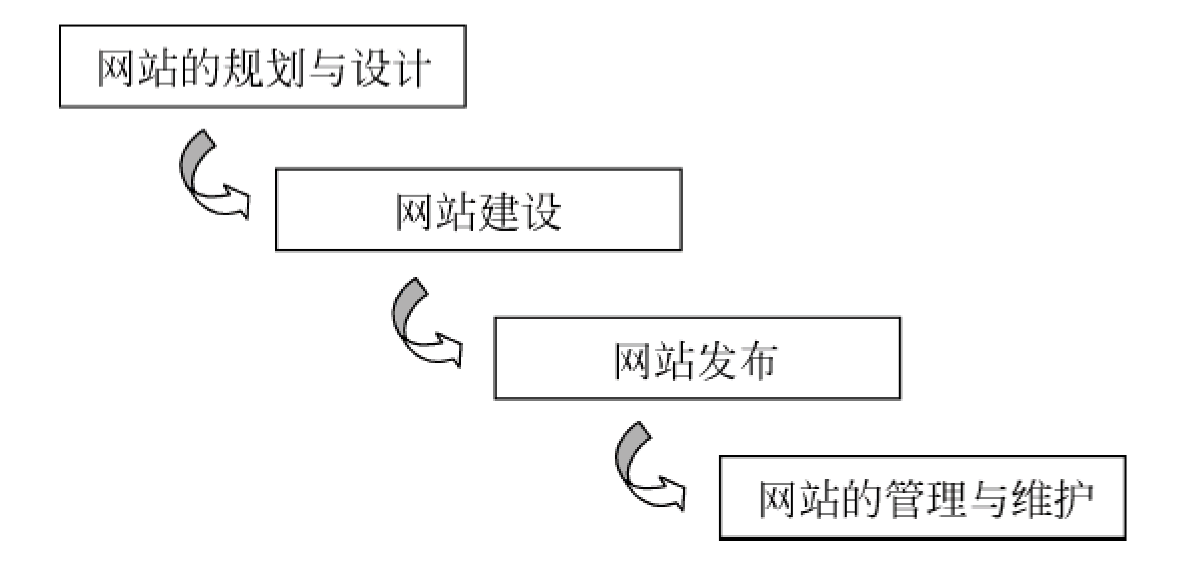


图 1.6　电子商务网站开发步骤

1.3.2　网站规划与设计

网站的规划与设计是网站建设的第一步。需要对网站进行一系列的分析和估计，然后根据分析的结果提出合理的建设方案，这就是网站的规划与设计。网站的规划与设计可分为网站定位、栏目规划、目录结构设计、风格设计、导航系统设计等环节。

1. 网站的定位

网站的定位就是确定网站的建设目标，它通常需要确定三个方面：网站的建设目的、网站的访问对象和网站的内容与服务。用更通俗的话来说，就是回答“为什么要建立这个网站？这个网站为谁服务？网站提供哪些方面的内容和服务？”

电子商务站点主要是为了在企业与企业、企业与个人消费者之间建立更为直接和高效的商务通道；电子政务网站代表的是政府部门，所以主要是通过它来宣传政府的形象、实现政务公开、向社会提供有价值的公益和导向信息，以及实现网上政务；个人站点则主要是为了介绍个人的兴趣爱好，通过共享信息来结识更多的朋友。

2. 栏目规划及其任务

栏目规划的主要任务是对所收集的大量内容进行有效的筛选，并将它们组织成一个合理的便于理解的逻辑结构，即建立网站的逻辑结构，在这其中不仅需要为整个网站建立层次型结构，还需要为每一个栏目或子栏目设计合理的逻辑结构。除此之外，栏目规划还需要确定哪些是重点栏目、哪些是需要实时更新的栏目、需要提供哪些功能性栏目等。

例如，对于一个企业网站来说，公司简介、产品介绍、服务内容、联系方式、技术支持等栏目都是必需的栏目。其中，产品介绍、价格信息、网上定购和产品动态等便是重点栏目。在确定必需的栏目和重点栏目后，再建立栏目的层次型结构，即从上到下一级一级地确定每一层的栏目，所有的栏目及其子栏目连在一起就形成了网站的层次型结构，常见的与线型结构、层次型结构和网络型结构。

3. 网站风格设计

风格设计包含很多内容，为了体现个人风格，符合网站的名称和定位，风格设计主要有网页色彩搭配和版面布局等。

网站的色彩是最影响网站整体风格的因素，也是站点美工设计中最令人头疼的问题。网站的色彩搭配通常分为两个步骤，那么第一步就是为整个网站选取一种主色调，然后再为主色调搭配多种适合的颜色。主色调指的是整个网站给人印象最深的颜色，或者说除白色之外用得最多的颜色。

版面布局设计需要根据内容文字的多少而自然分割，同报纸、杂志一样，网站也分为很多不同的网页，如主页、栏目首页和内容网页等。

4. 网站目录结构设计

目录结构又称为物理结构，解决的是如何在硬盘上更好地存放包括网页文件、图片和数据库文件等各种资源在内的所有网站资源问题。

目录结构是否合理，对网站的创建效率会产生较大的影响，但更主要的是，会对未来网站的性能、网站的维护及扩展产生很大的影响。

不建议将所有的网页文件和资源文件都放在同一个目录下。当文件较多时，Web 服务器的性能就会急剧下降，因为查找一个网页文件需要很长的时间，而且网站管理员在区分不同性

质的文件和查找某一个特定的文件时也会变得非常麻烦。

通常，根据栏目规划来设计目录结构，将图片及资源文件都放在一个独立的 images 目录下，目录的层次不要太深，不要使用中文目录名，可执行文件和不可执行文件分开放置，数据库文件单独放置。

5. 网站的导航设计

在访问网站时也一样，用户期望在任何一个网页上都能清楚地知道目前所处的位置，并且能快速地从这个网页切换到另一网页。因此网站导航对于一个网站来说非常的必要和重要，它是衡量一个网站是否优秀的重要标准。

导航最常用的实现方法就是导航条。在导航条中，所有超链接对应的网页在网站的层次型结构中是并列的，所以通过它可以快速地切换到并列的其他网页。如图 1.7 所示的腾讯新闻网首页(网站为 http://news.qq.com)中就有很多导航条。首先是网站第一层分类栏目的导航条，这个导航条几乎出现在腾讯新闻网的所有网页中，所以在任何一个网页通过它都可以立即跳转到国内、国际和社会等各个栏目的首页。



图 1.7　腾讯网首页

1.3.3　网站建设

在网站的规划与设计完成之后，就可以着手进行站点的建设工作了。站点建设分为 4 部分，分别为域名注册、网站平台配置、网页制作和网站测试。

1. 域名注册

域名注册是将网站的域名合法化以便用户能通过这个域名访问电子商务网站。注册之后别人就不能再次使用相同的域名进行注册了。

2. 网站平台配置

网站配置首先要做的就是为网站选择合适的发布平台，即选择适合网站规模的各种软硬

件资源,包括确定服务器解决方案和软件平台等。

常见的服务器解决方案如下。

- 自建服务器:易于控制,安全。但需要申请专线接入,适用于处理敏感数据的站点,如电子商务站点。
- 托管服务器:将自己的服务器放到高带宽入网(100M 以上)的电信局或专门的数据中心,通过拨号、ISDN 或 DDN 等低速线路远程维护。该方案提供优越的机架空间、网络安全防护、UPS 供电、恒温恒湿环境及防火设施等,适用于中、大规模的网站。许多大型网站就是以此形式建立站点及镜像站点的。
- 租用服务器:与托管服务器相似,只是主机不是自己的。
- 虚拟主机:在同一计算机硬件、同一操作系统上,可以建立多个 Web 站点,每个站点在访问者看来好像是在一个独立的主机硬件上,这样的 Web 站点称为虚拟主机。每一个虚拟主机具有独立的域名,可以共享一个 IP 地址,也可以具有独立的 IP 地址,能提供完整的 Internet 服务(包括 Web、FTP、E-mail 等)。其缺点是,不支持高访问量,远程管理权限有限,软件安装不方便。它适用于小企业做产品宣传和业余爱好者发布数据。

常见的软件平台如下。

- 操作系统软件:UNIX、Linux 和 Windows 操作系统等。
- Web 服务器软件:IIS、Apache 和 AIX 等。
- 数据库管理系统:DB2、Oracle、Sybase、SQL Server 和 MySQL 等。

3. 网页制作

网页制作指的就是使用网页制作工具来制作每一个网页。网页文件是一种特殊的文本文件,所以它的制作工具非常广泛,如记事本和 Ultraedit 等都可以用来制作网页。但是最为主要的还是使用专用的网页制作工具,如 Macromedia 公司开发的 Dreamweaver 系列、微软公司开发的 FrontPage 和 Visual Studio 等。

4. 网站测试

所有网页都制作完成之后,在正式对外发布网站之前,还有一步非常重要的工作就是网站测试。网站测试的目的就是为了保证在网站发布之后所有的用户都能正常地浏览网页和使用所提供的服务。

网站测试通常包括功能测试、性能测试、可用性测试、客户端兼容性测试和安全性测试等 5 个方面。

1.3.4　网站发布和网站的管理与维护

当网页基本制作完毕、网站测试基本通过之后,即可发布网站,以便让所有的因特网用户都能通过因特网访问这个网站。网站发布最基本的工作就是将网页传送到 Web 服务器之上,最常用的方式就是使用 FTP。FTP 是因特网中用于访问远程机器的一个协议,使用户可以在本地机和远程机之间进行有关文件的操作,允许传输任意文件并且允许文件具有所有权与访问权限。也就是说,通过 FTP 协议,可以与因特网上的 FTP 服务器进行文件的上传或下载等动作,可以实现发布网站的功能。另外,一些强大的网页制作工具如 Visual Studio 也具有网页发布的功能。

网站管理与维护是一项非常繁重的工作，目的是为了保证网站的正常运行。网站管理与维护的内容主要分为安全管理、性能管理和内容管理等方面。

1.4　练　习　题

1. 单项选择题

(1) Internet 是目前全世界规模最大、信息资源最多的计算机网络，它是一个(　　)。

A. 外部网　　B. 专用网　　C. 公共信息网　　D. 城域网

(2) 目前，人们所提及的电子商务多指在(　　)上开展的商务活动。

A. 手机　　B. 网络　　C. 计算机　　D. POS 机

(3) 电子商务是在(　　)技术与网络通信技术的互动发展中产生和不断完善的。

A. 电子　　B. 微波　　C. 多媒体　　D. 计算机

(4) 电子商务的主要流通渠道是(　　)。

A. 企业-批发商-零售商-消费者　　B. 企业-消费者

C. 企业-中介商-消费者　　D. 企业-零售商-消费者

(5) 电子商务的主要成分是(　　)。

A. 电子　　B. 商务　　C. 交易　　D. 网络

(6) 电子商务的前提是(　　)。

A. 商务信息化　　B. 商务国际化　　C. 交易国际化　　D. 交易网络化

(7) 对电子商务的理解，至少应从(　　)两个方面考虑。

A. 计算机技术和商务　　B. 现代信息技术和商务

C. 网络技术和商务　　D. 通信技术和商务

(8) TCP/IP 协议是(　　)上所使用的协议。

A. Internet　　B. Intranet　　C. Extranet　　D. LAN

(9) 网页是 WWW 的基本文档，它是用(　　)编写的。

A. HTML 语言　　B. C 语言

C. FORTRAN 语言　　D. Java 语言

(10) 以下(　　)不属于 Web 网页开发技术。

A. ASP .NET　　B. Linux　　C. JSP　　D. PHP

(11) 以下文件扩展名中只有(　　)不是静态网页的后缀。

A. html　　B. shtml　　C. htm　　D. aspx

(12) 以下文件名后缀中只有(　　)是静态网页的后缀。

A. asp　　B. aspx　　C. htm　　D. jsp

(13) 下列说法正确的是(　　)。

A. 页面上有动态的东西就是动态网页

B. 静态网页内容固定，交互性能比动态网页差

C. ASP、JSP 和 ASP .NET 技术都是把脚本语言嵌入到 HTML 文档中

D. ASP .NET 程序和 ASP 程序一样都是解释执行

(14) HTTP 协议是一种(　　)协议。

A. 文件传输协议　　B. 远程登录协议

C. 邮件协议　　D. 超文本传输协议

2. 问答题

(1) 什么是电子商务？和传统商务比较，电子商务在哪些方面有所不同？

(2) 电子商务有哪些主要的分类方法？

(3) 简述电子商务网站开发的主要步骤。

(4) 简述静态网页和动态网页执行时的最大区别。

第 2 章　电子商务网站开发环境配置



本章指南

- 电子商务网站开发环境
- 安装和配置Visual Studio 2012
- 安装和使用SQL Server 2012

2.1　电子商务网站开发环境

知识梳理

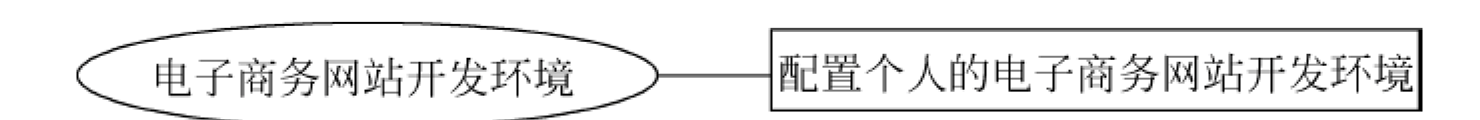


图 2.1 所示是典型的 ASP .NET 电子商务网站开发环境。客户机与 Web 服务器之间以及 Web 服务器与数据库服务器之间可以通过局域网或 Internet 进行通信。

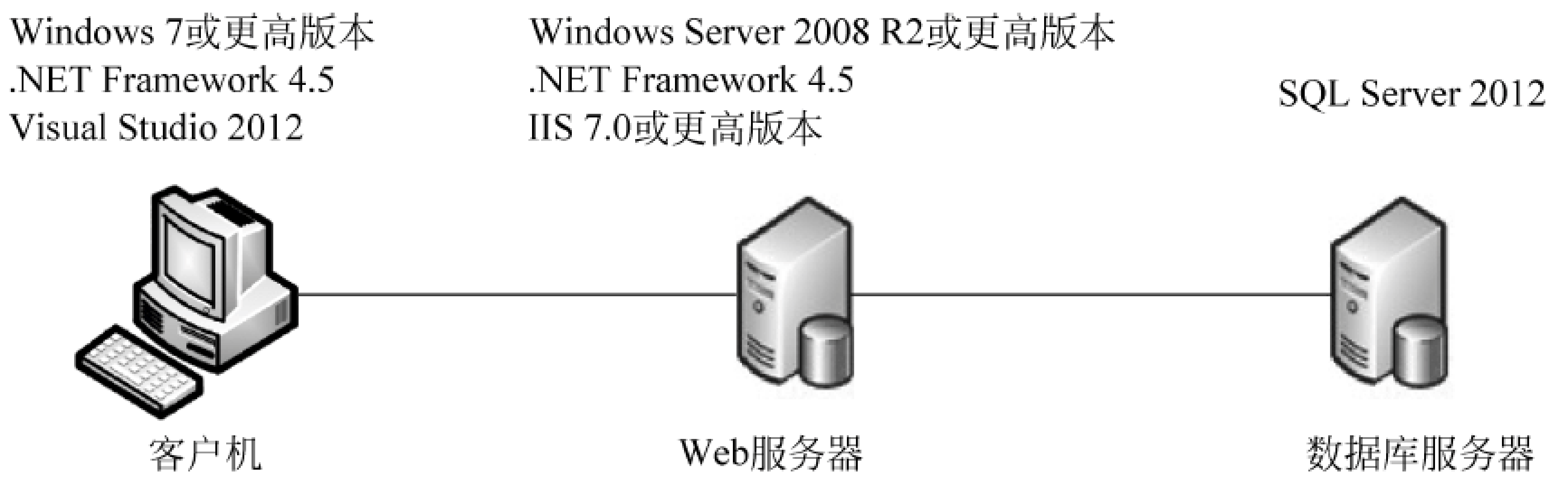


图 2.1　电子商务网站开发环境

上述 Web 开发环境简称为 Windows＋IIS＋ASP .NET＋SQL Server。采用的开发工具为 Visual Studio。

Visual Studio 2012 是微软公司于 2012 年 9 月推出的基于 Windows 的开发环境，支持包括 ASP .NET Web、WPF 和 WinForms 客户端、Silverlight、平行计算和云计算等众多用于开发，还在核心集成开发环境（IDE）、代码编辑器、编程语言，以及企业设计、架构师、Office 集成、SharePoint 3.0 集成和测试工具方面做了显著的改进，是目前电子商务网站的优秀平台。

.NET Framework 是 .NET 应用程序开发框架的运行库，也就是说如果执行的程序是用 .NET 开发的，就需要 .NET Framework 作为底层运行环境。在开发 .NET 应用程序时，也需要 .NET Framework 的支持。

Visual Studio 2012 基于 .NET Framework 4.5，可以开发 Windows 窗体应用程序、Web 应用程序和 Web 服务等，并支持 C#、VB、C++ 和 F# 等多种计算机语言。

IIS（Internet Information Services）是用于创建、管理和承载 ASP .NET 网站的 Web 服务器，它是 Windows 的一个组件。为了方便 Web 开发，Visual Studio 2012 自带了 IIS Express，它是一款免费软件，且小型、轻量特别适合 ASP .NET 开发人员使用的 Web 开发服务器。IIS Express 除了没有 IIS 的可视化管理界面，几乎用于其全部的功能。

SQL Server 2012 是微软公司推出的新一代数据库管理系统，不仅延续现有数据平台的强大能力，全面支持云技术，并且能够快速构建相应的解决方案，实现云之间数据的扩展与应用的迁移，包括支持来自于不同网络环境的数据的交互，全面的自助分析等创新功能。针对大数据以及数据仓库，SQL Server 2012 提供从几 TB 到数百 TB 全面端到端的解决方案。

如果在本机上开发电子商务网站（也称为独立开发方式），采用的本地计算机既是服务器又是客户机，需要配置客户端软件，如 Windows 7 或更高版本（带有浏览器）；又要配置服务端软件，如 .NET Framework 4.5、Visual Studio 2012、IIS Express（Visual Studio 内置的轻量级的 IIS）和 SQL Server（如果网页需要访问 SQL Server 数据库）等。本书的开发实例采用独立开发方式，如图 2.2 所示。

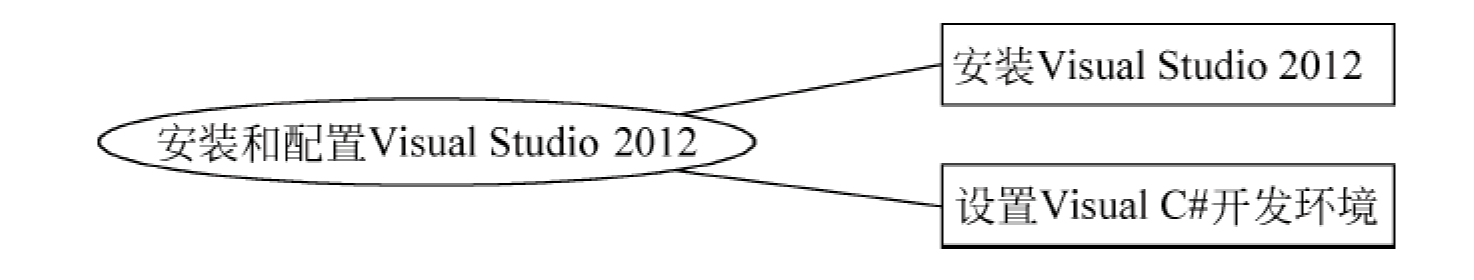


图 2.2　ASP .NET 网站的独立开发方式

在开发好的电子商务网站部署到 Web 服务器上后，客户机只需要安装 Windows 7（带 IE 浏览器），便可以浏览该网站。

2.2　安装和配置 Visual Studio 2012

知识梳理



2.2.1　安装 Visual Studio 2012

1. 下载和安装 .NET Framework 4.5

首先安装 .NET Framework 4.5。如果计算机上没有安装 .NET Framework 4.5，可以从微软网站免费下载，其网址为 http://www.microsoft.com/zh-cn/download/details.aspx?id=

30653,下载的文件为 dotNetFx45_Full_setup. exe。安装步骤如下:

① 双击 dotNetFx45_Full_setup 应用程序,出现“ .NET Framework 4.5 安装程序”对话框,勾选“我已阅读并接受许可条款”复选框,如图 2.3 所示,单击“安装”按钮。

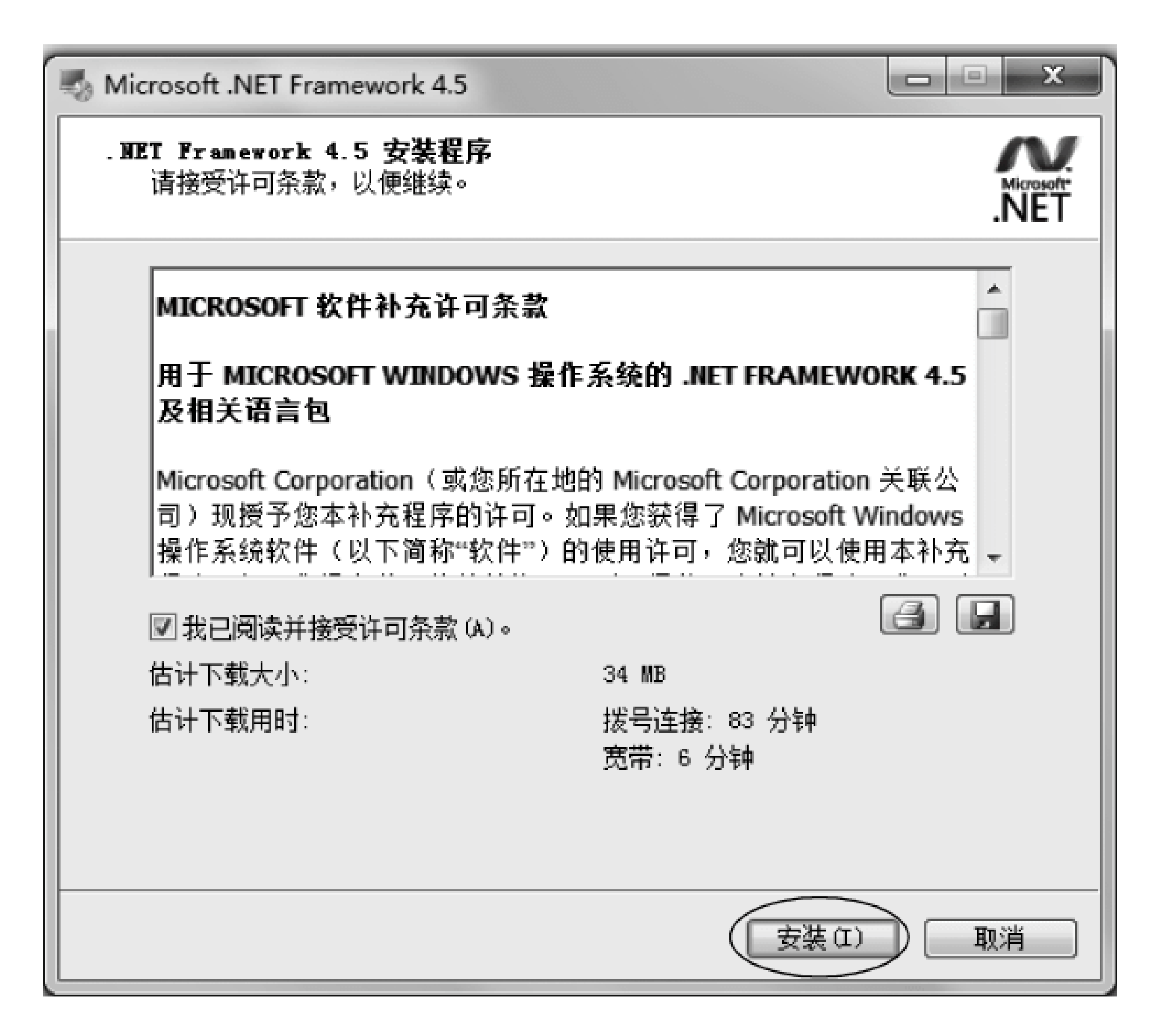


图 2.3 “ .NET Framework 4.5 安装程序”对话框

② 系统开始安装 .NET Framework 4.5,如图 2.4 所示。安装完成后出现如图 2.5 所示的“安装完毕”页面,单击“完成”按钮。



图 2.4 “安装进度”页面

在安装好 .NET Framework 4.5 后,选择“开始|控制面板|程序和功能”命令,看到如图 2.6 所示的“卸载或更改程序”对话框,表明 .NET Framework 4.5 安装成功。

2. 下载和安装 Visual Studio 2012

如果计算机上没有安装 Visual Studio 2012,可以从微软公司网站免费下载,其网址为



图 2.5　“安装完毕”页面



图 2.6　“卸载或更改程序”对话框

http://www.microsoft.com/zh-cn/download/details.aspx? id=30682，进入该网页后，单击“下载”按钮，勾选 VS2012_PRO_chs.iso 文件，单击 Next 按钮，将这些文件下载到本地计算机上。安装步骤如下：

① 双击 VS2012_PRO_chs.iso 文件，解压到 VS2012_PRO_chs 目录中。

② 进入该目录，双击执行 vs_professional 应用程序。出现安装提示对话框，勾选“我同意许可条款和条件”复选框，如图 2.7 所示，单击“下一步”按钮。

③ 出现如图 2.8 所示的对话框，单击“安装”按钮。

④ 系统开始安装，出现如图 2.9 所示的提示信息。



图 2.7　安装提示一

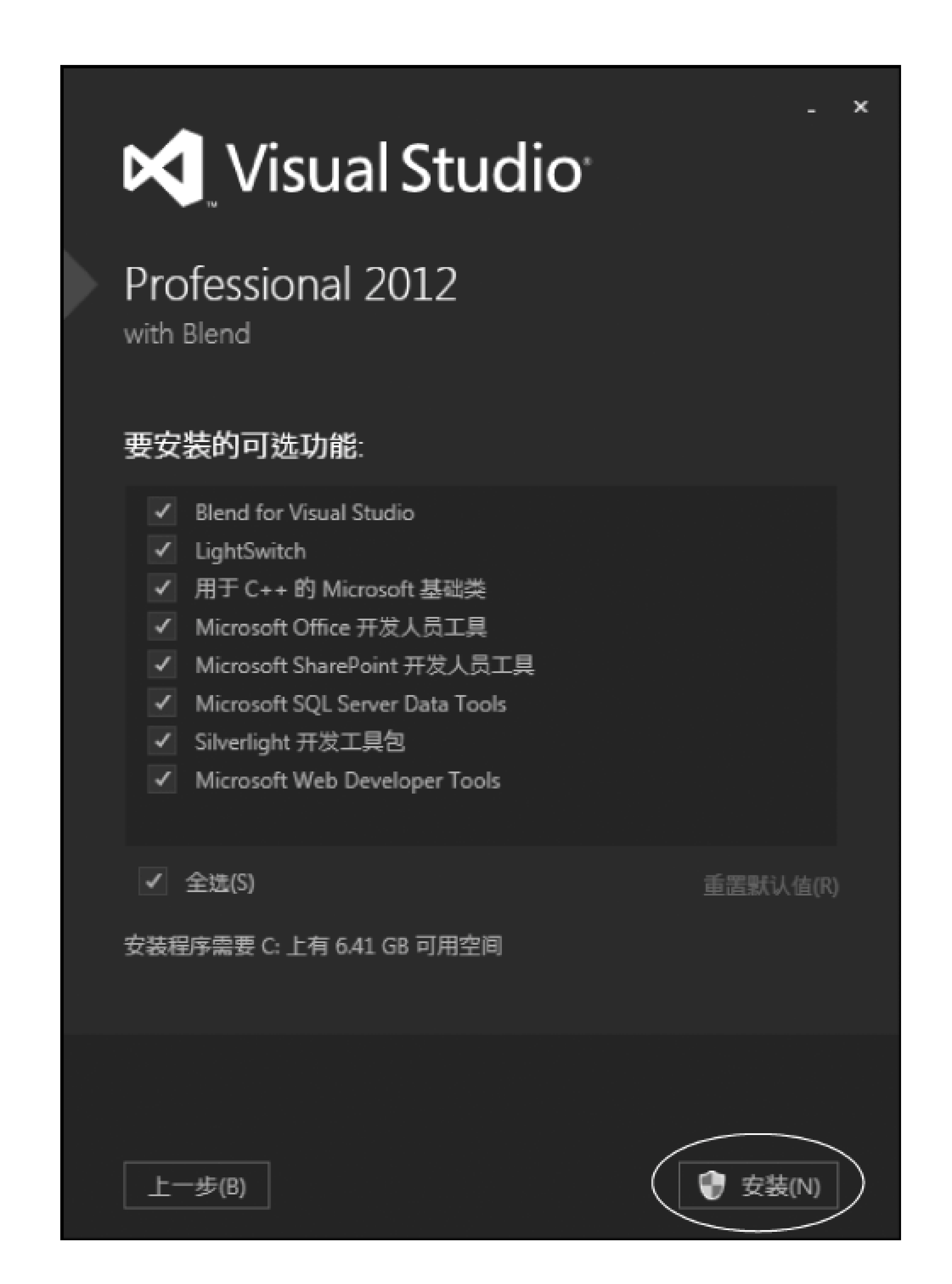


图 2.8　安装提示二

⑤ 安装完成后出现如图 2.10 所示的页面，单击“启动”按钮可以启动 Visual Studio 2012 系统。



图 2.9　安装提示三



图 2.10　安装完成

2.2.2　设置 Visual C# 开发环境

Visual Studio 2012 支持多种语言开发，为了支持 C# 语言，首先要将 Visual Studio 2012 配置成 Visual C# 开发环境。

在安装 Visual Studio 2012 后，若当前不是 C# 开发环境，设置为 C# 开发环境的操作步骤如下：

① 选择"开始|所有程序|Microsoft Visual Studio 2012|Visual Studio 2012"命令启动 Visual Studio 2012 系统。

② 在 Visual Studio 2012 的集成开发环境中选择"工具|导入导出设置"命令，出现如图 2.11 所示的欢迎使用设置向导对话框，单击"下一步"按钮。

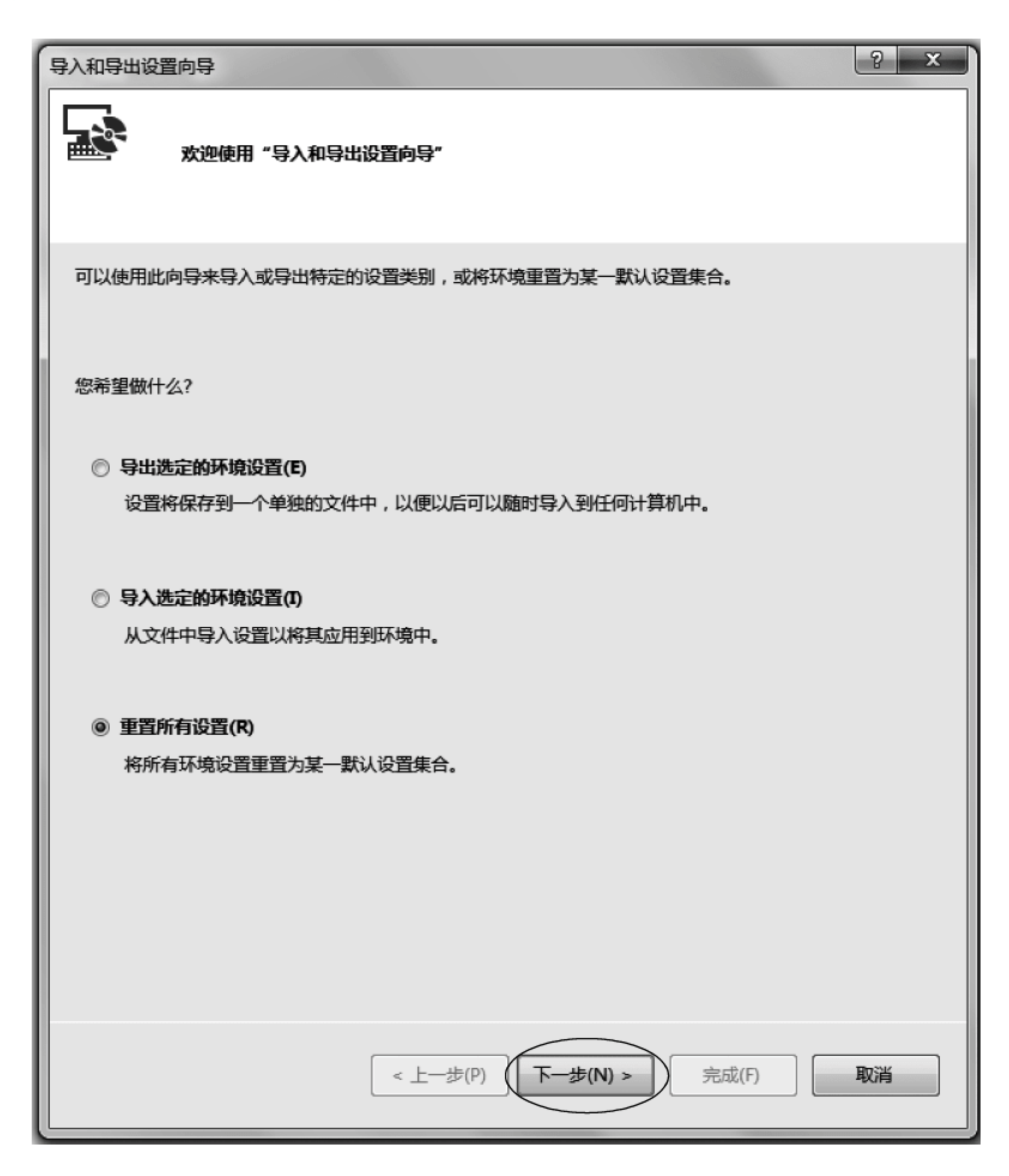


图 2.11　欢迎使用设置向导对话框

③ 出现如图 2.12 所示的"保存当前设置"页面，保持所有默认设置，单击"下一步"按钮。

④ 出现"选择一个默认设置集合"页面，单击"Visual C# 开发设置"选项，如图 2.13 所示。单击"完成"按钮即可完成配置。



图 2.12　“保存当前设置”页面

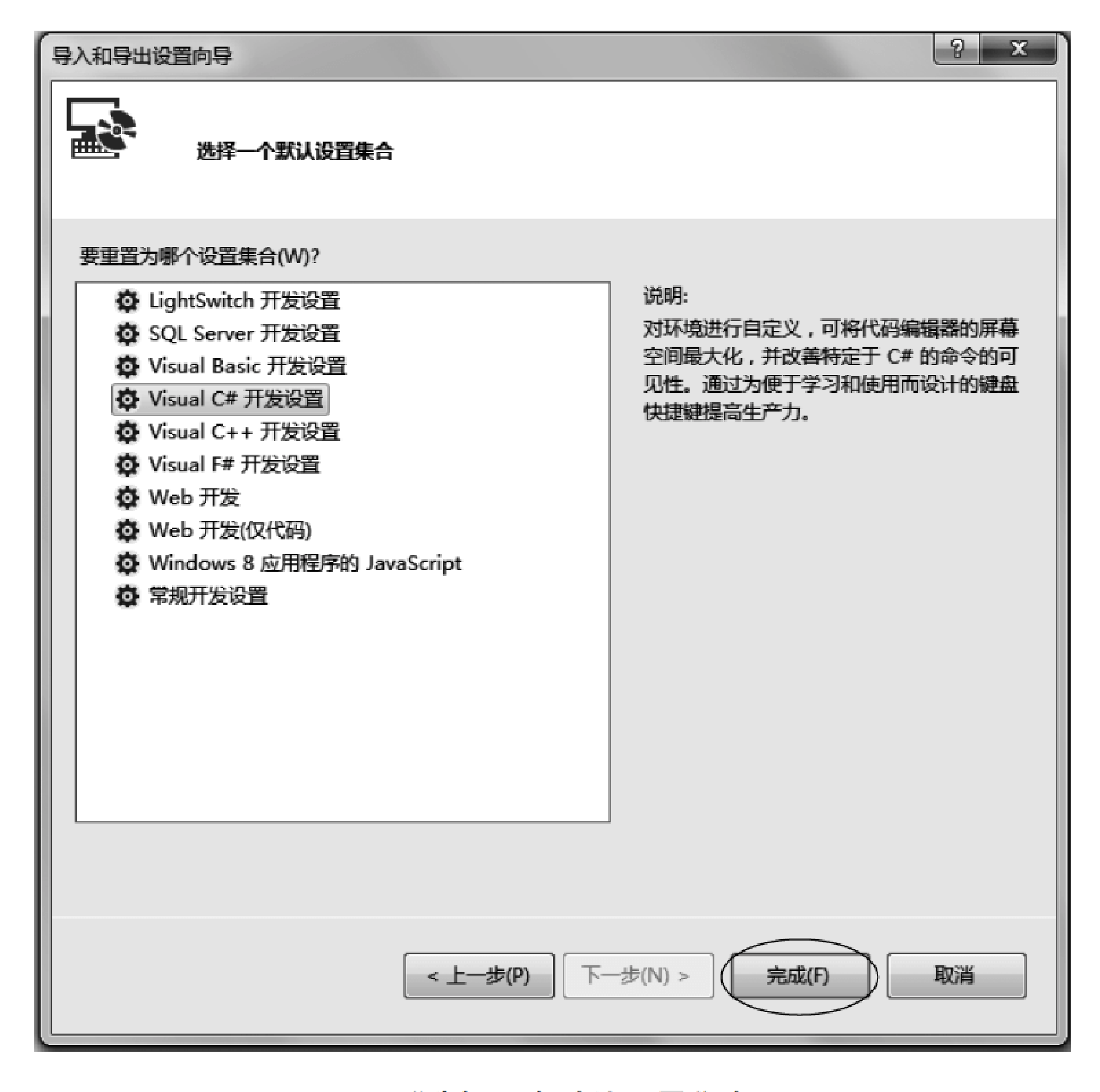
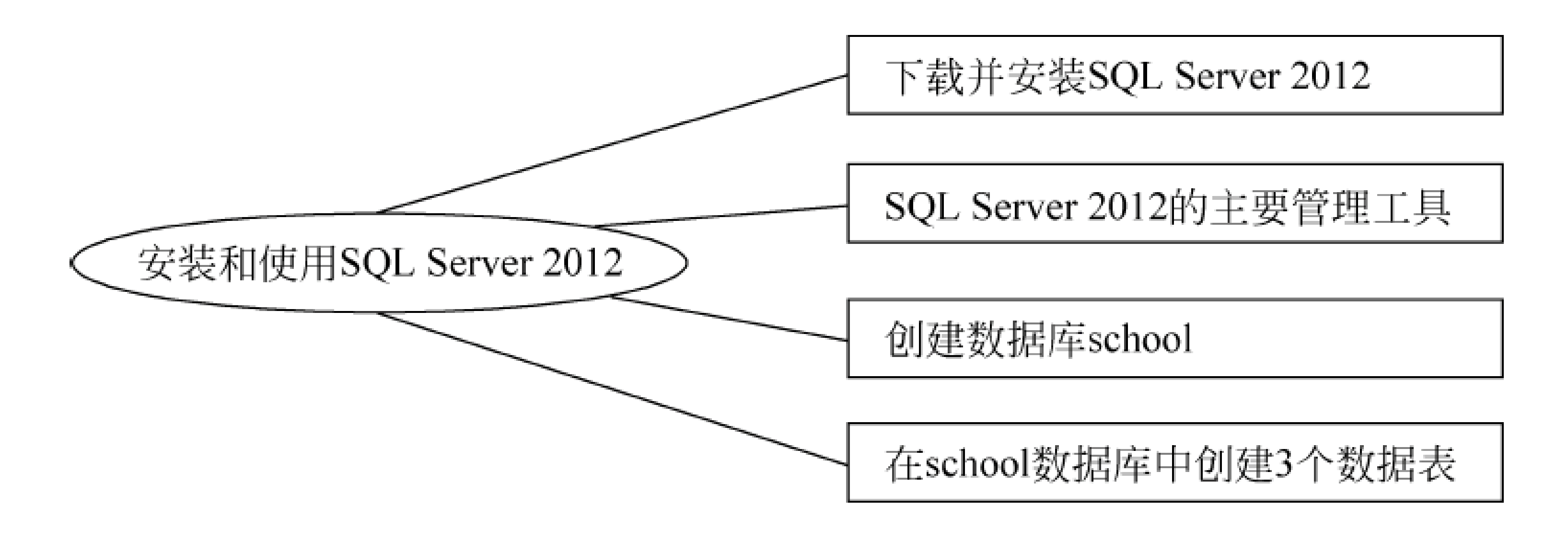


图 2.13　“选择一个默认设置集合”页面

2.3　安装和使用 SQL Server 2012

知识梳理



2.3.1　下载并安装 SQL Server 2012

首先确定用户的计算机系统是几位的,可以右击桌面上的“计算机”,在出现的快捷菜单中选择“属性”命令,例如看到如图 2.14 所示的结果,表示计算机系统是 32 位系统的。



图 2.14　Windows 的“属性”

然后从微软(中国)网站下载 SQL Server 2012 Express 免费版本。下载网址为 http://www.microsoft.com/zh-cn/download/。Microsoft®SQL Server®2012 Express 既包含 32 位版本,也包含 64 位版本。这里下载 32 位版本,下载 SQLEXPR_x86_CHS.exe 文件存放在自己的文件夹中。安装步骤如下:

① 双击 SQLEXPR_x86_CHS.exe 文件解压到 SQLFULL_x86_CHS 文件夹,单击其中的 SETUP 应用程序。出现如图 2.15 所示的安装初始界面。单击左侧的“安装”项,出现如图 2.16 所示的“安装”页面,单击“全新 SQL Server 独立安装或向现有安装添加功能”项开始独立安装过程。

② 首先进入“安装程序支持规则”页面,如图 2.17 所示,表明通过了检查。单击“下一步”按钮。进入“产品密钥”页面,输入 SQL Server 企业服务器版的序列号,如图 2.18 所示。

③ 单击“下一步”按钮,出现“许可条款”对话框,勾选“我接受许可条款”项,如图 2.19 所示,单击“下一步”按钮。出现“设置角色”页面,选择“SQL Server 功能安装”单选按钮,如图 2.20 所示,单击“下一步”按钮。出现“功能选择”页面,单击“全选”按钮,如图 2.21 所示,单击“下一步”按钮。进入“安装规则”页面,如图 2.22 所示,表明通过了检查,单击“下一步”按钮。

说明:数据库引擎是 SQL Server 的核心组件,它作为服务运行在服务器上,通常被称为 SQL Server 实例。在一台服务器上可以同时运行多个 SQL Server 实例。



图 2.15　安装初始界面



图 2.16　“安装”页面



图 2.17　“安装程序支持规则”页面

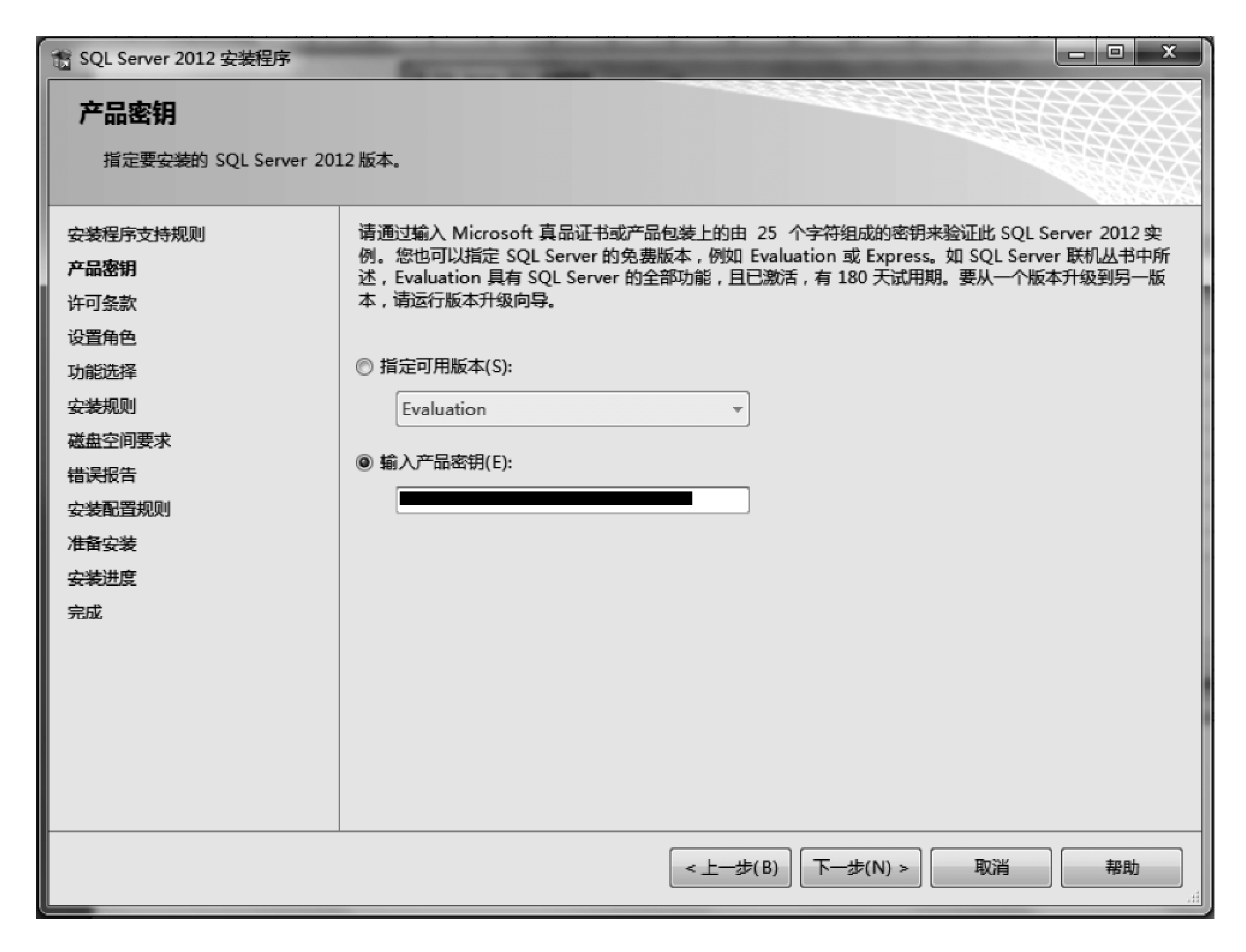


图 2.18　“产品密钥”页面

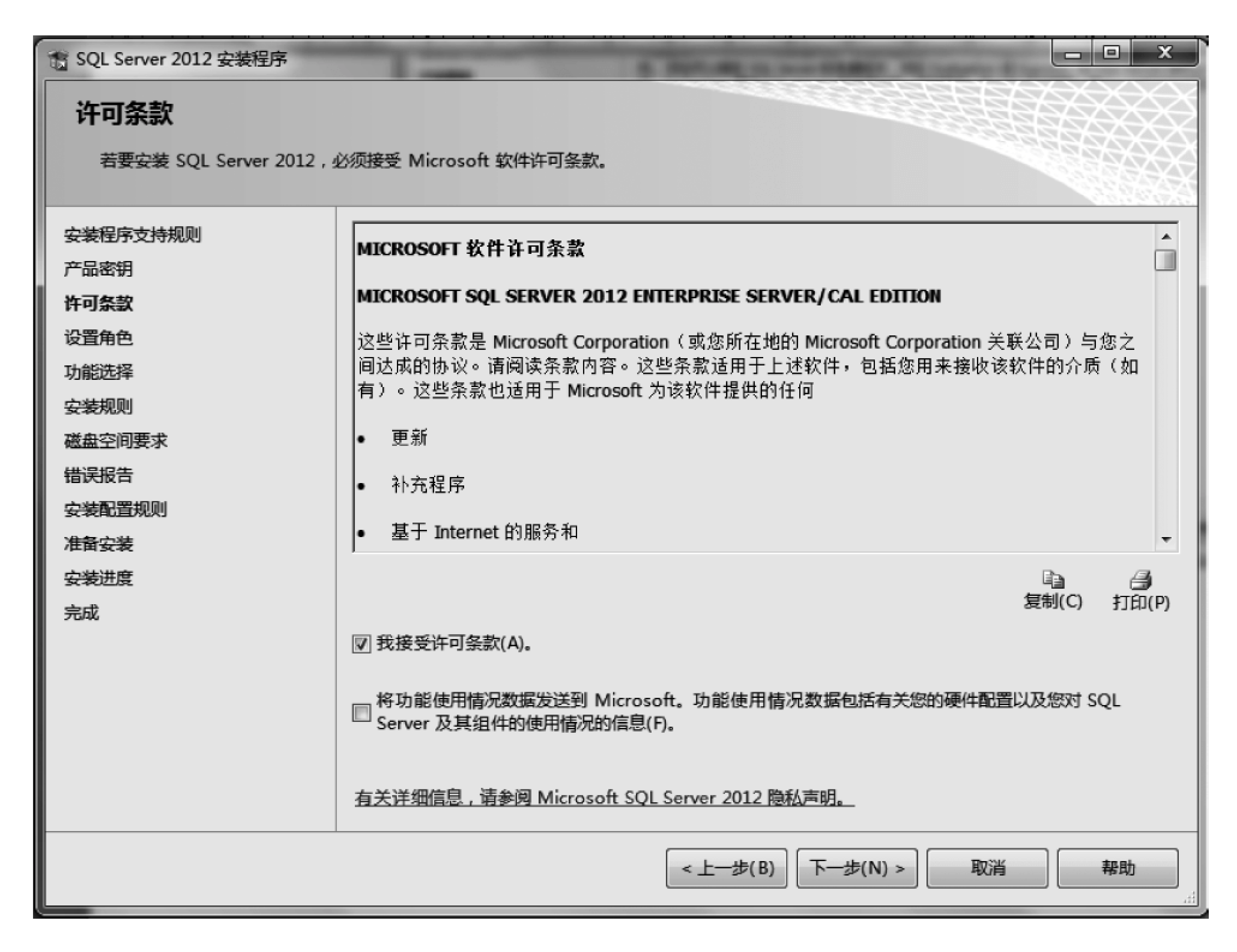


图 2.19 “许可条款”页面

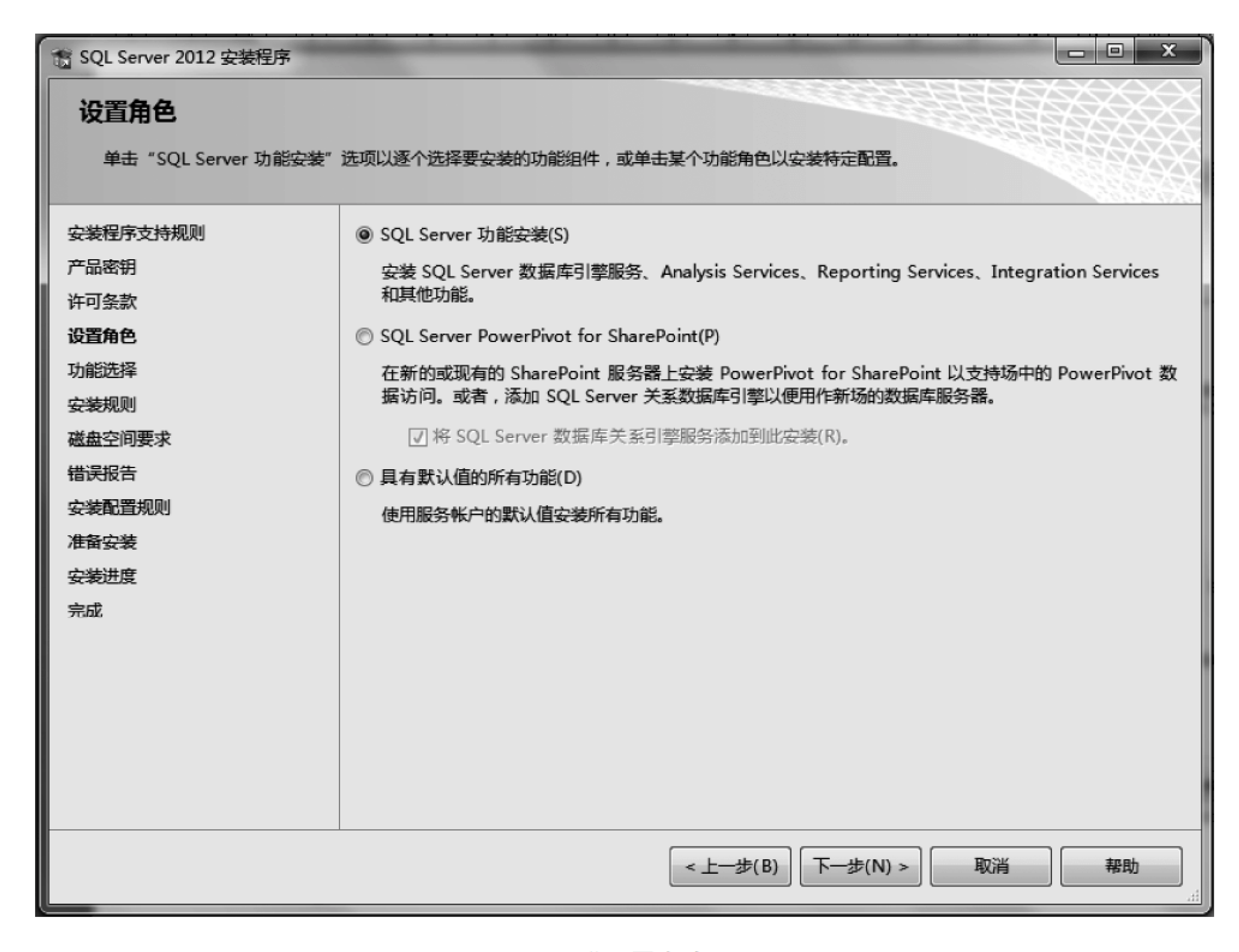


图 2.20 “设置角色”页面

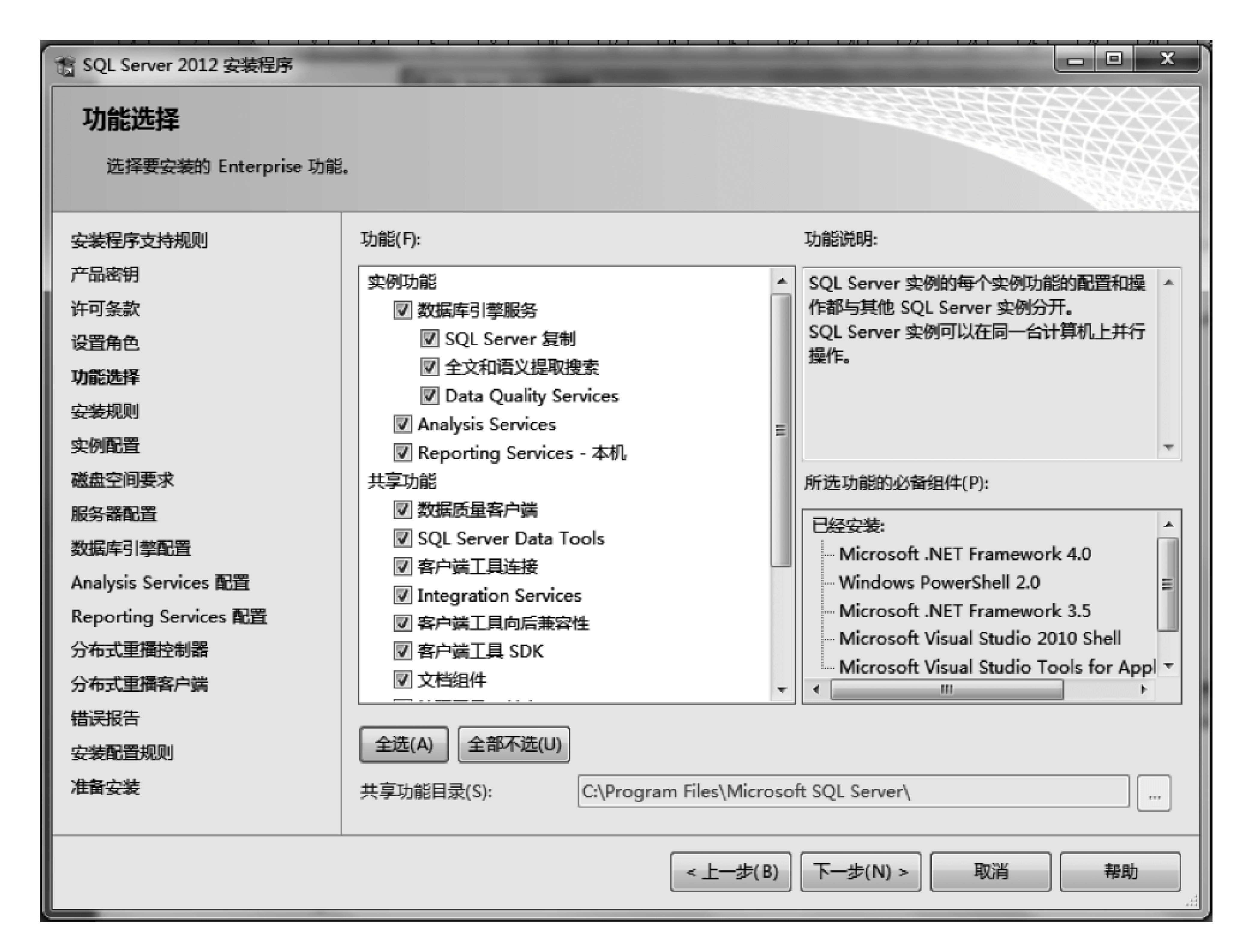


图 2.21　“功能选择”页面

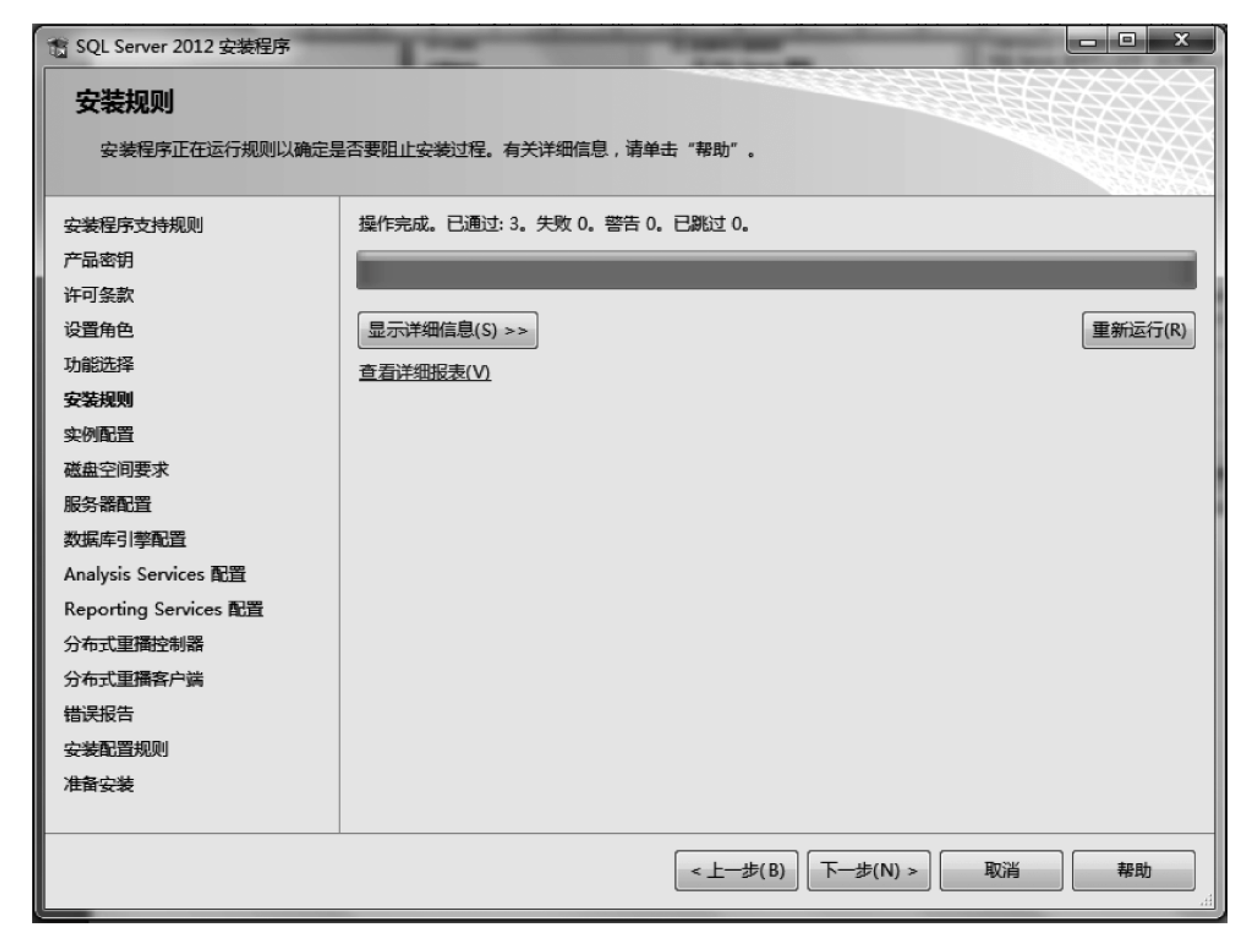


图 2.22　“安装规则”页面

④ 进入“实例配置”页面，如图 2.23 所示，保持默认项，即默认 SQL Server 实例名为 MSSQLSERVER，单击“下一步”按钮。出现“磁盘空间要求”页面，如果磁盘空间足够，直接单击“下一步”按钮。进入“服务器配置”页面，如图 2.24 所示，单击“下一步”按钮。

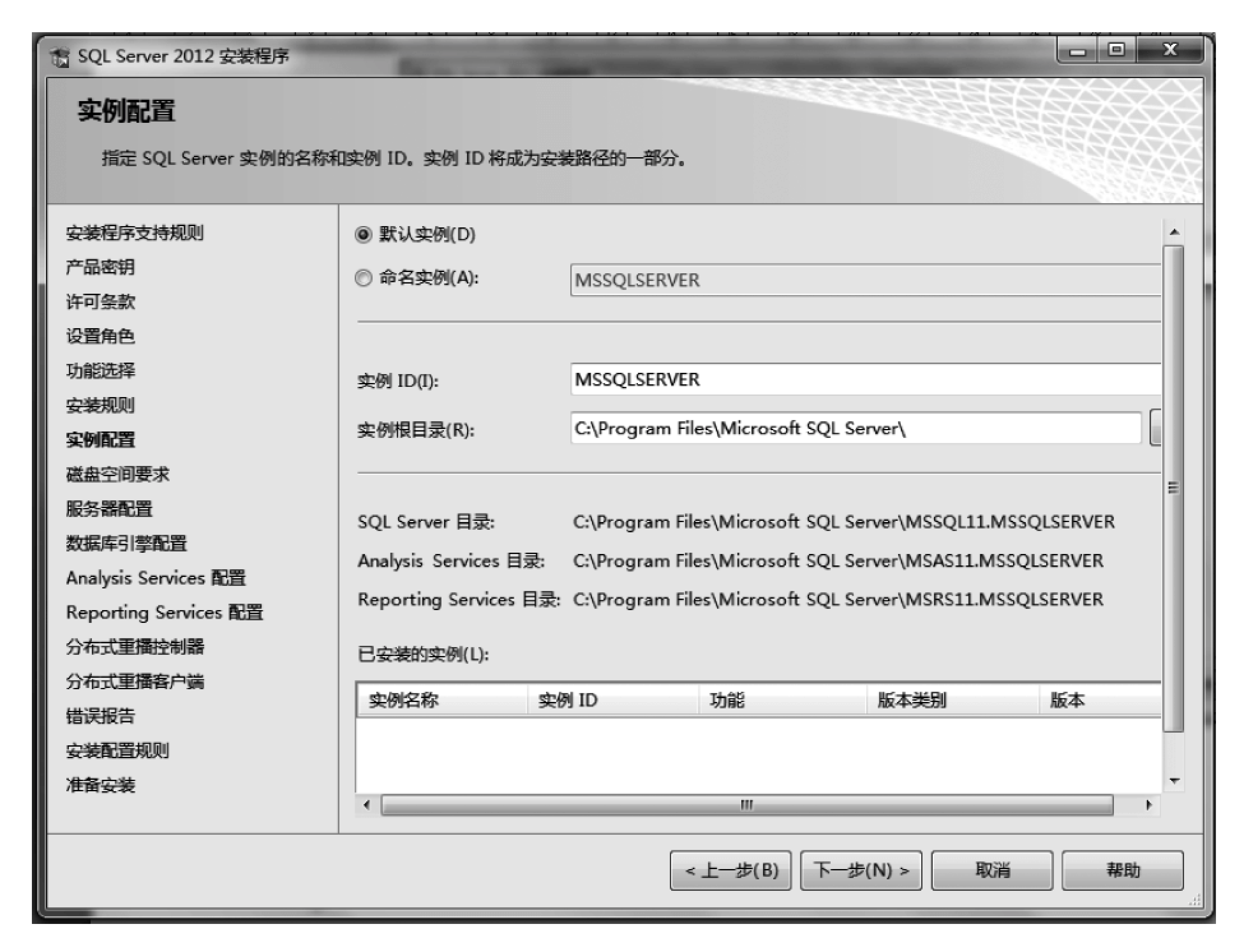


图 2.23　“实例配置”页面

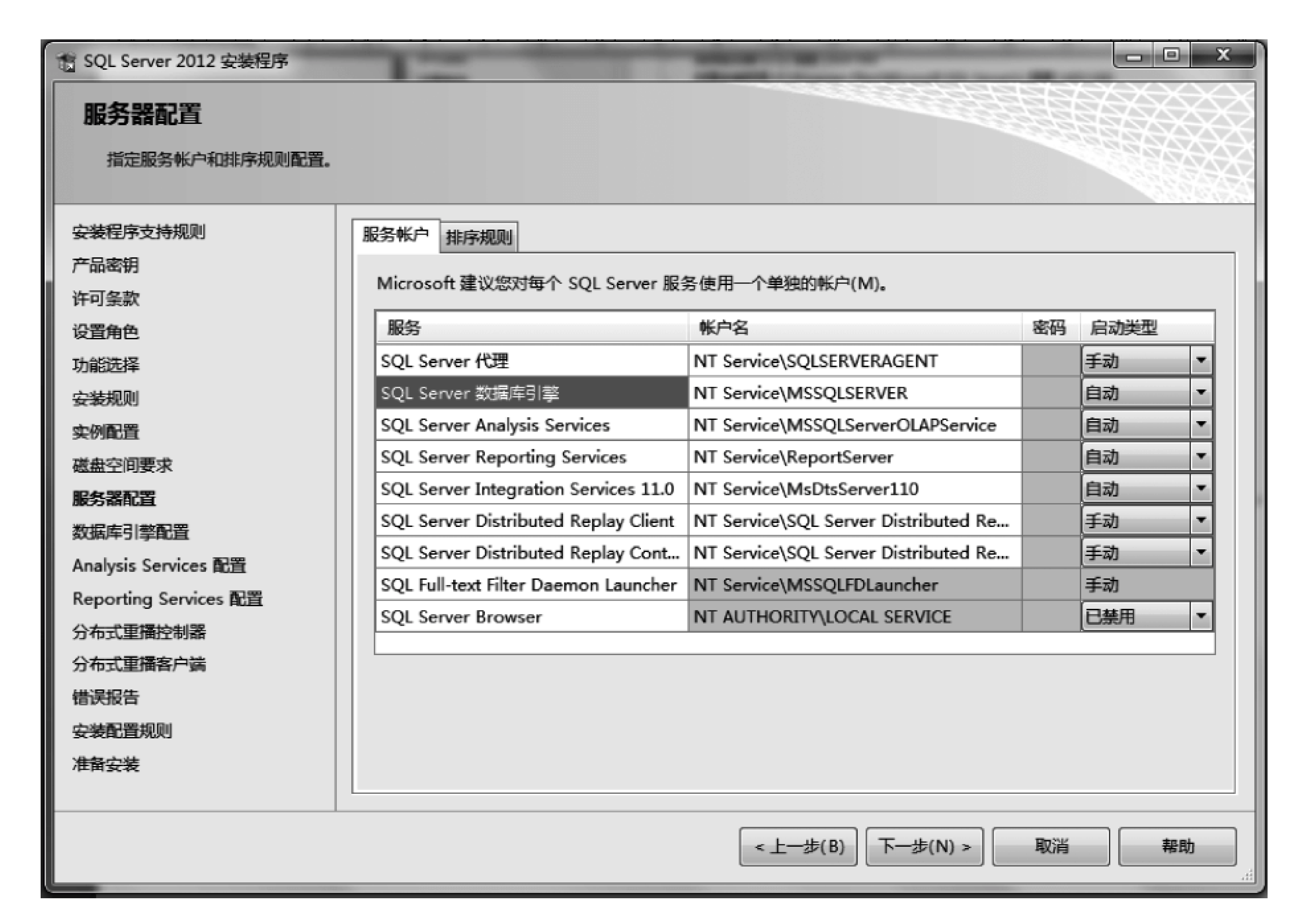


图 2.24　“服务器配置”页面

⑤ 进入“数据库引擎配置”页面，设置“身份验证模式”为“Windows 身份验证模式”。单击“添加当前用户”按钮添加 LCB-PC\Administrator（Administrator）管理员，再单击“添加”按钮，出现“选择用户或组”对话框，如图 2.25 所示。单击“高级”按钮，在出现的对话框中单击“立即查找”按钮，在“搜索结果”列表中选择 Administrator，如图 2.26 所示，单击两次“确定”按钮返回“数据库引擎配置”页面，如图 2.27 所示，看到指定的 SQL Server 管理员为 LCB-PC\Administrator（Administrator）和 Administrator（Administrator）。单击“下一步”按钮。



图 2.25　“选择用户或组”对话框一



图 2.26　“选择用户或组”对话框二

说明：本机的服务器名为 LCB-PC，管理员账号为 Administrator。为 SQL Server 配置的身份验证模式为“Windows 身份验证模式”。

⑥ 进入“Analysis Services 配置”页面，采用与数据库引擎配置相同的操作指定 Analysis Services 管理员为 LCB-PC\Administrator（Administrator）和 Administrator（Administrator），如图 2.28 所示。单击“下一步”按钮。



图 2.27　“数据库引擎配置”页面

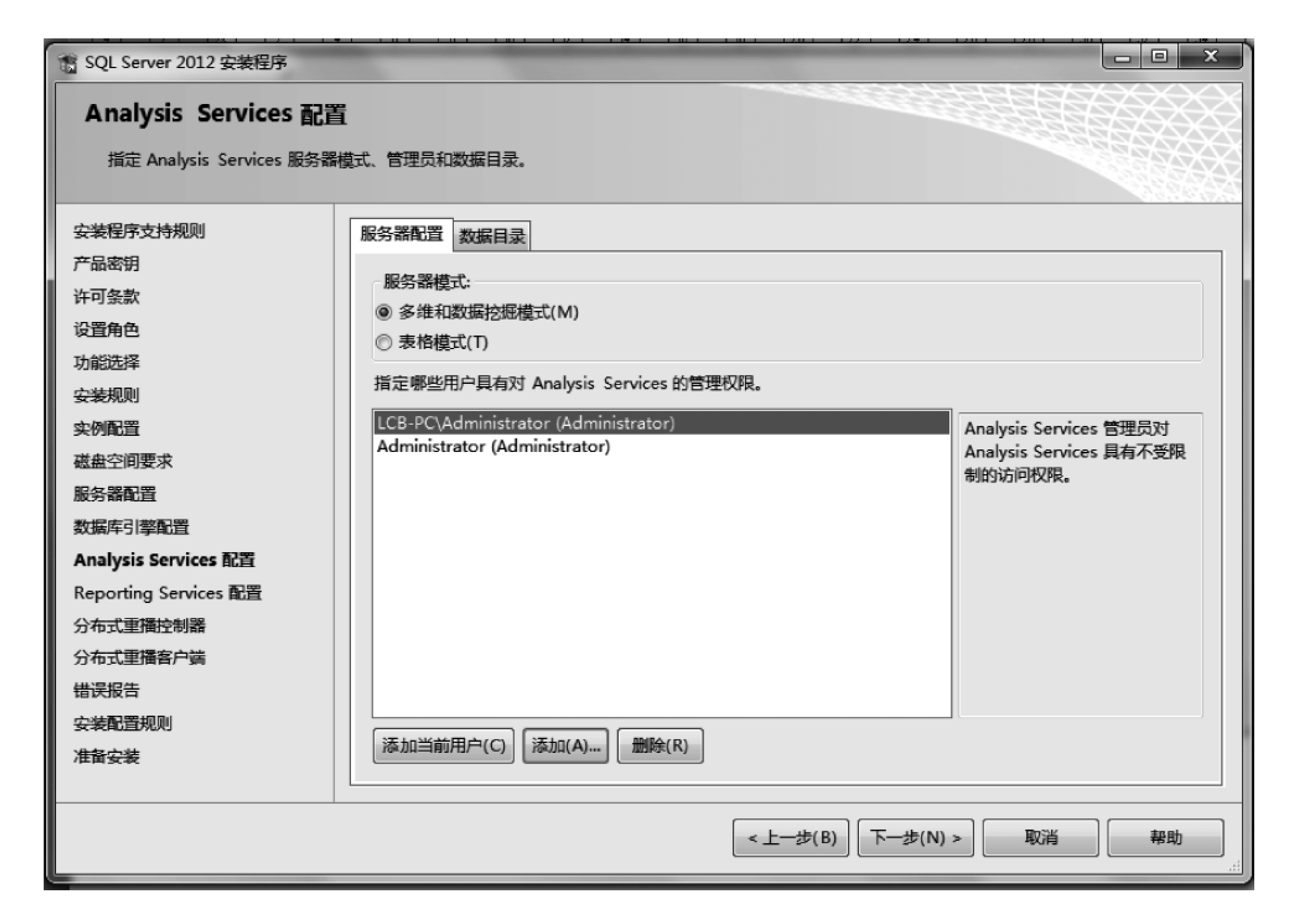


图 2.28　“Analysis Services 配置”页面

说明： Analysis Services(分析服务)是 SQL Server 的重要组件，为商业智能应用程序提供了联机分析处理(OLAP)功能和各种数据挖掘算法，开发人员可以设计、创建和管理包含从其他数据源(如关系型数据库)聚合来的数据的多维结构，以实现各种数据挖掘。

⑦ 进入"Reporting Services 配置"页面，选择"仅安装"单选按钮，如图 2.29 所示，单击"下一步"按钮。

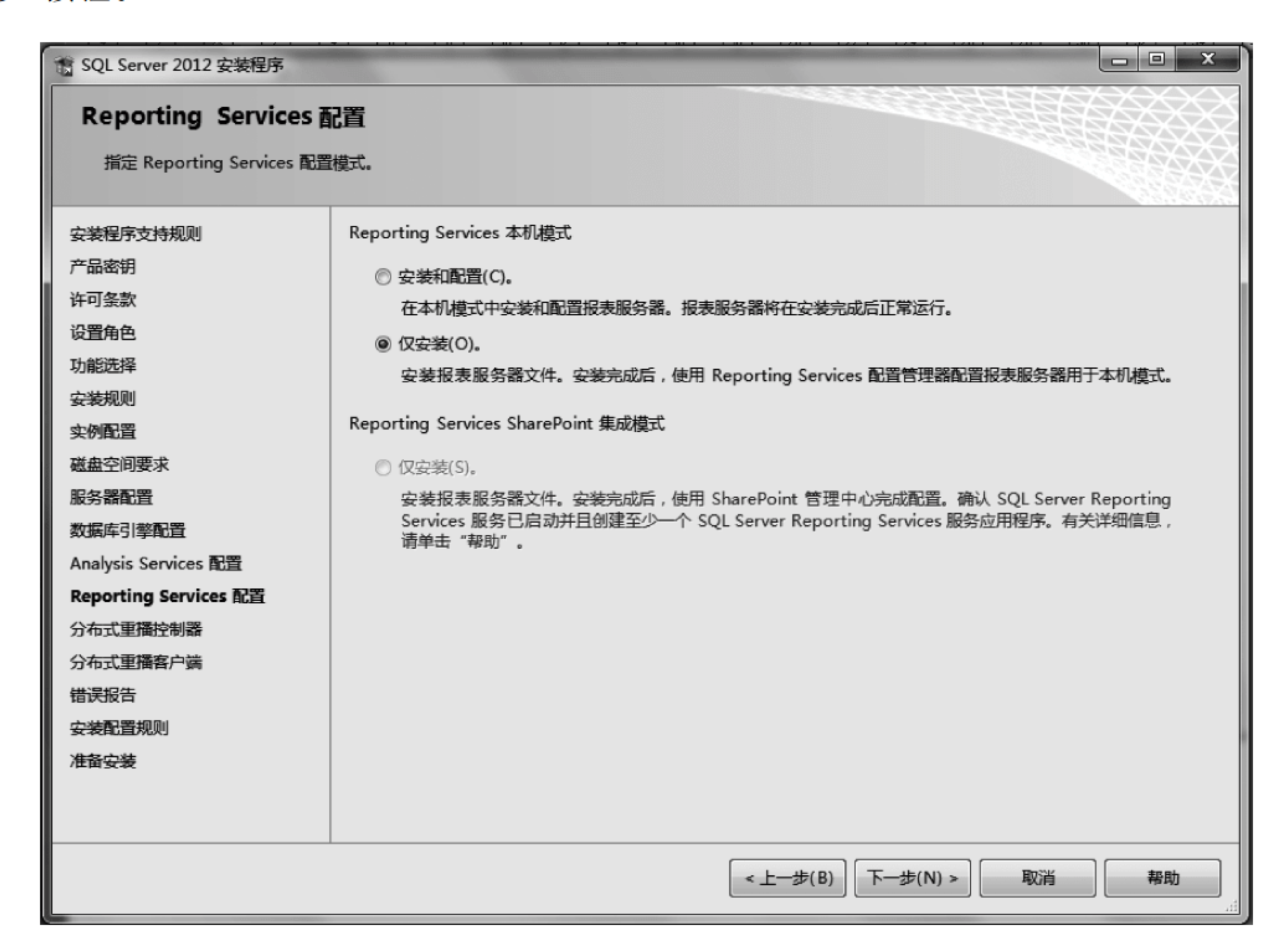


图 2.29　"Reporting Services 配置"页面

说明： Reporting Services 即报表服务，它是一种基于服务器的新型报表平台，可用于创建和管理包含来自关系数据源和多维数据源数据的表格报表、矩阵报表、图形报表和自由格式报表。可以通过基于 Web 的连接来查看和管理创建的报表。

⑧ 进入"分布式重播控制器"页面，采用与数据库引擎配置相同的操作指定 Analysis Services 管理员为 LCB-PC\Administrator(Administrator)和 Administrator(Administrator)，如图 2.30 所示。单击"下一步"按钮，出现"分布式重播客户端"页面，保持默认项，单击"下一步"按钮。

⑨ 出现"错误报告"页面，保持默认项，如图 2.31 所示，单击"下一步"按钮。出现"安装配置规则"页面，如图 2.32 所示，表明安装配置规则检查已通过，单击"下一步"按钮。

⑩ 出现"准备安装"页面，如图 2.33 所示，单击"安装"按钮。系统开始安装 SQL Server 文件，显示如图 2.34 所示的"安装进度"页面。安装完毕，显示如图 2.35 所示的"完成"页面，单击"关闭"按钮。

说明： 采用上述步骤安装 SQL Server 2012 后，本地计算机上既包含有 SQL Server 服务器工具，也包含有 SQL Server 客户端开发工具。

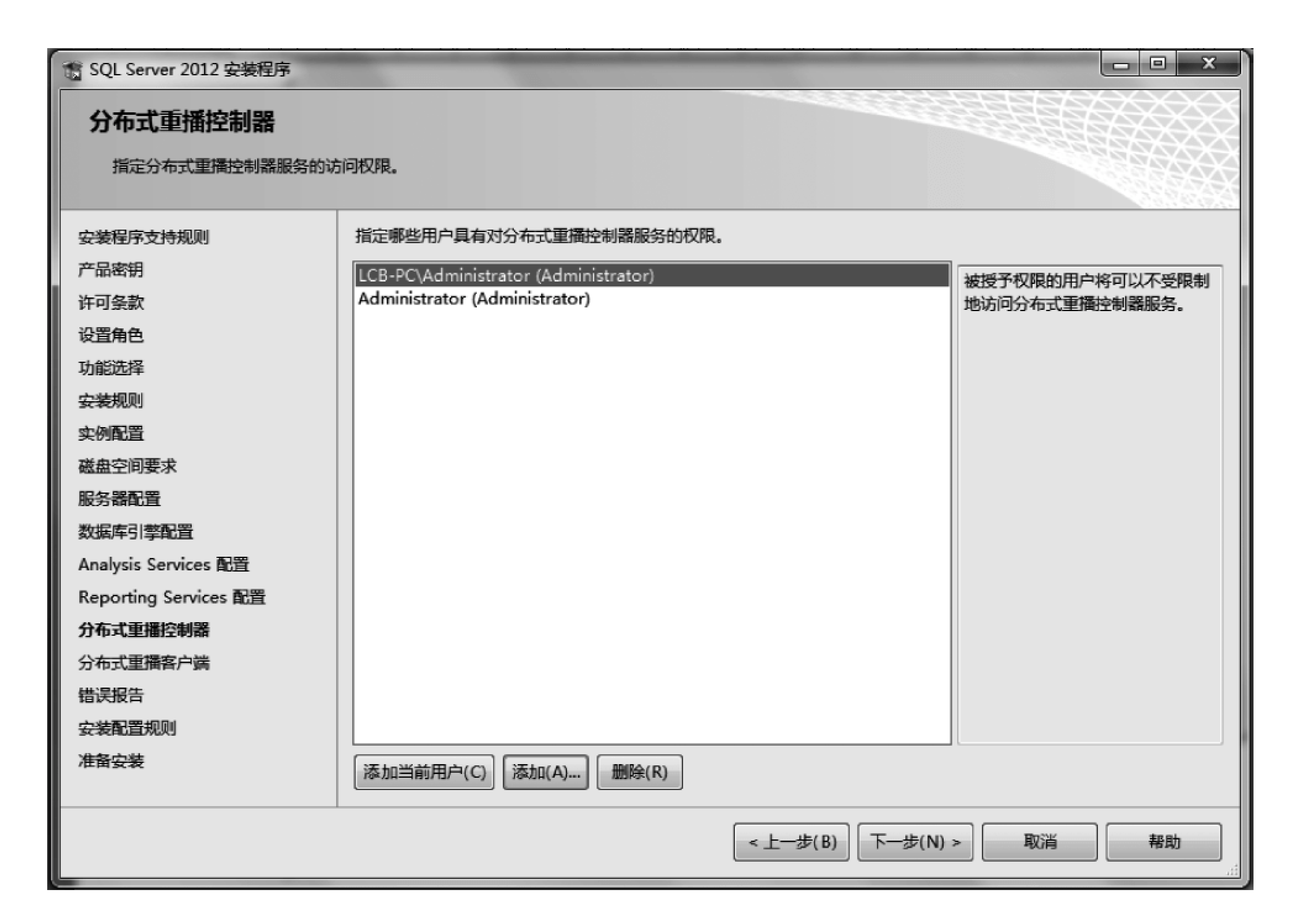


图 2.30 “分布式重播控制器”页面

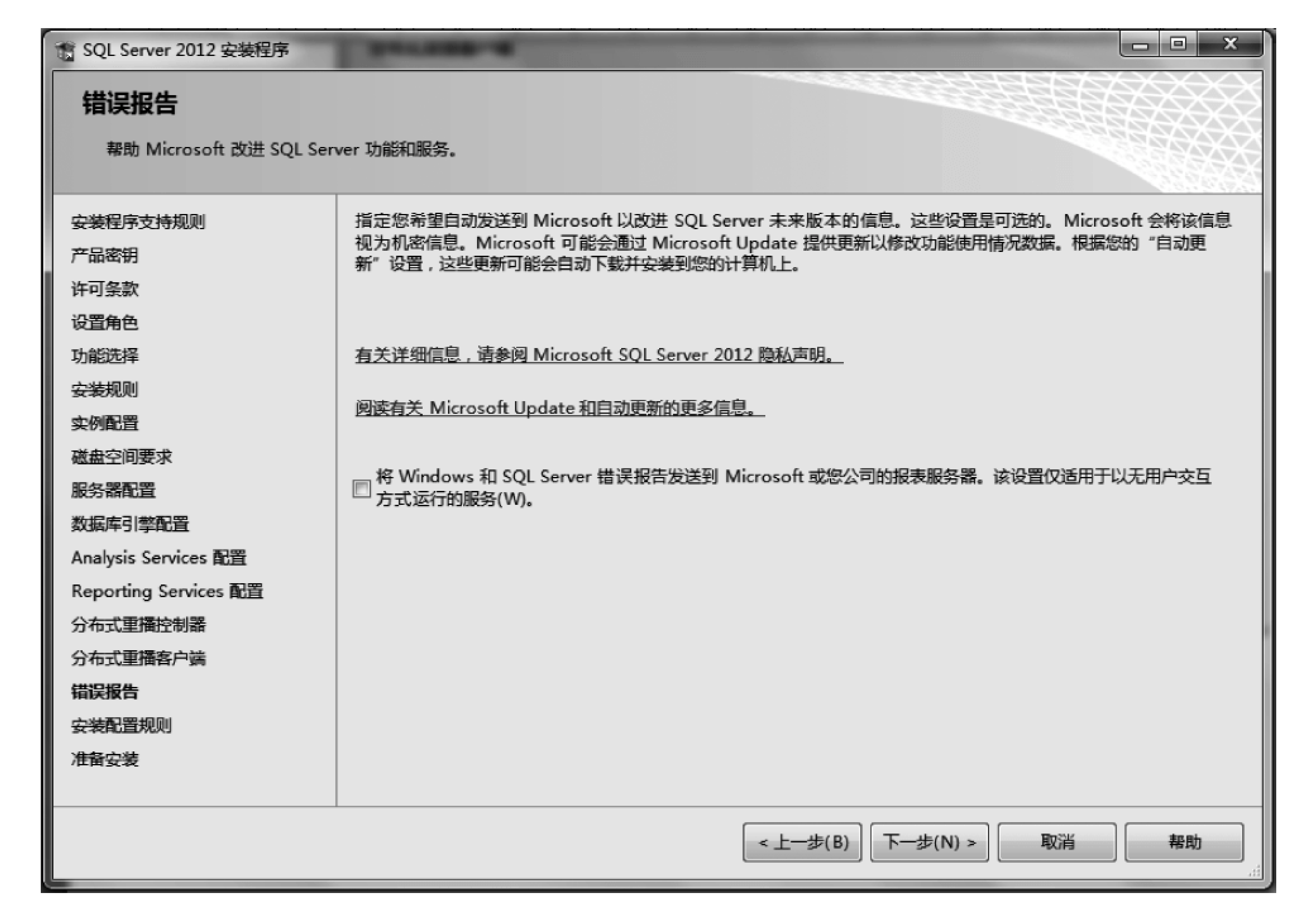


图 2.31 “错误报告”页面



图 2.32　“安装配置规则”页面

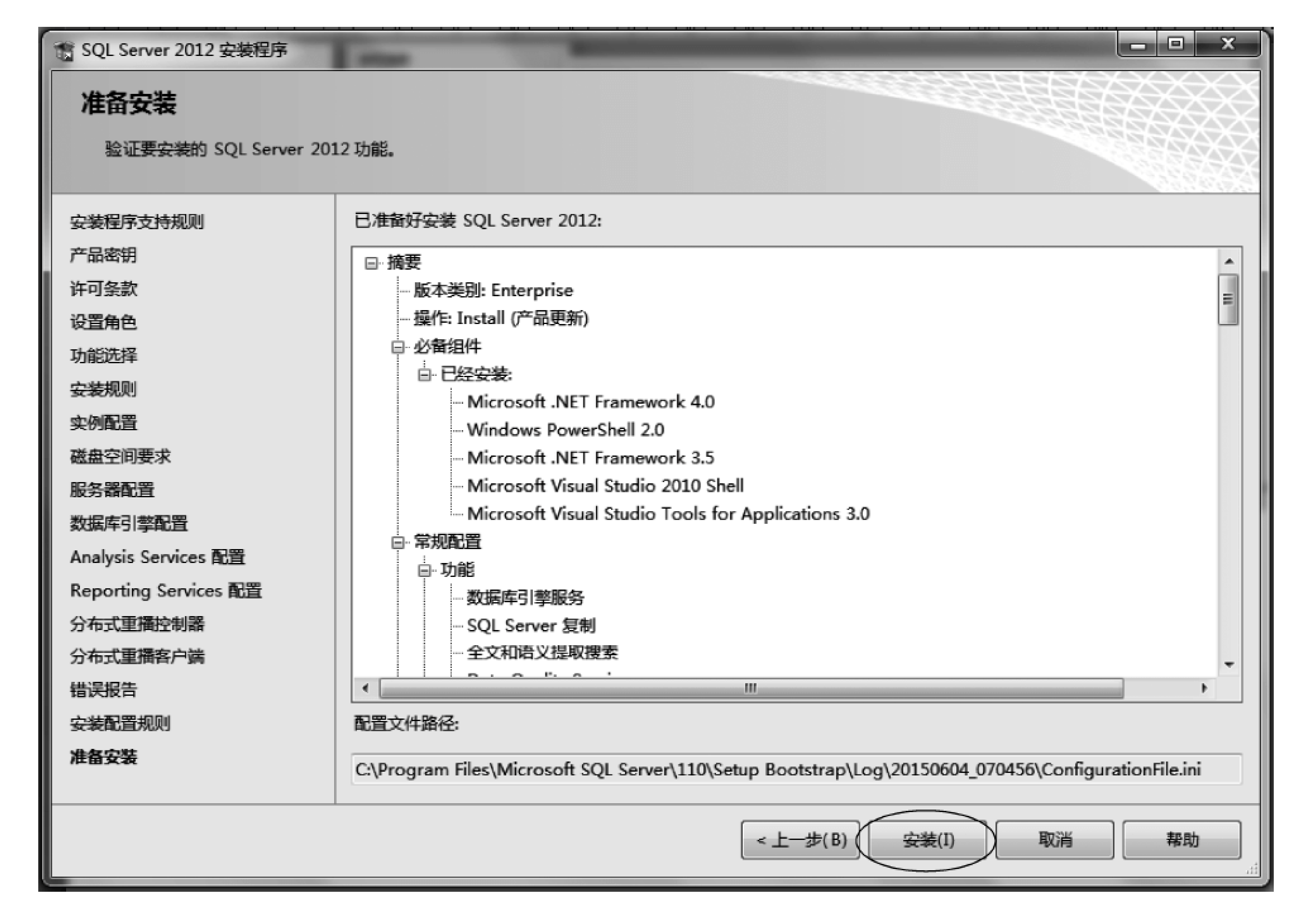


图 2.33　“准备安装”页面

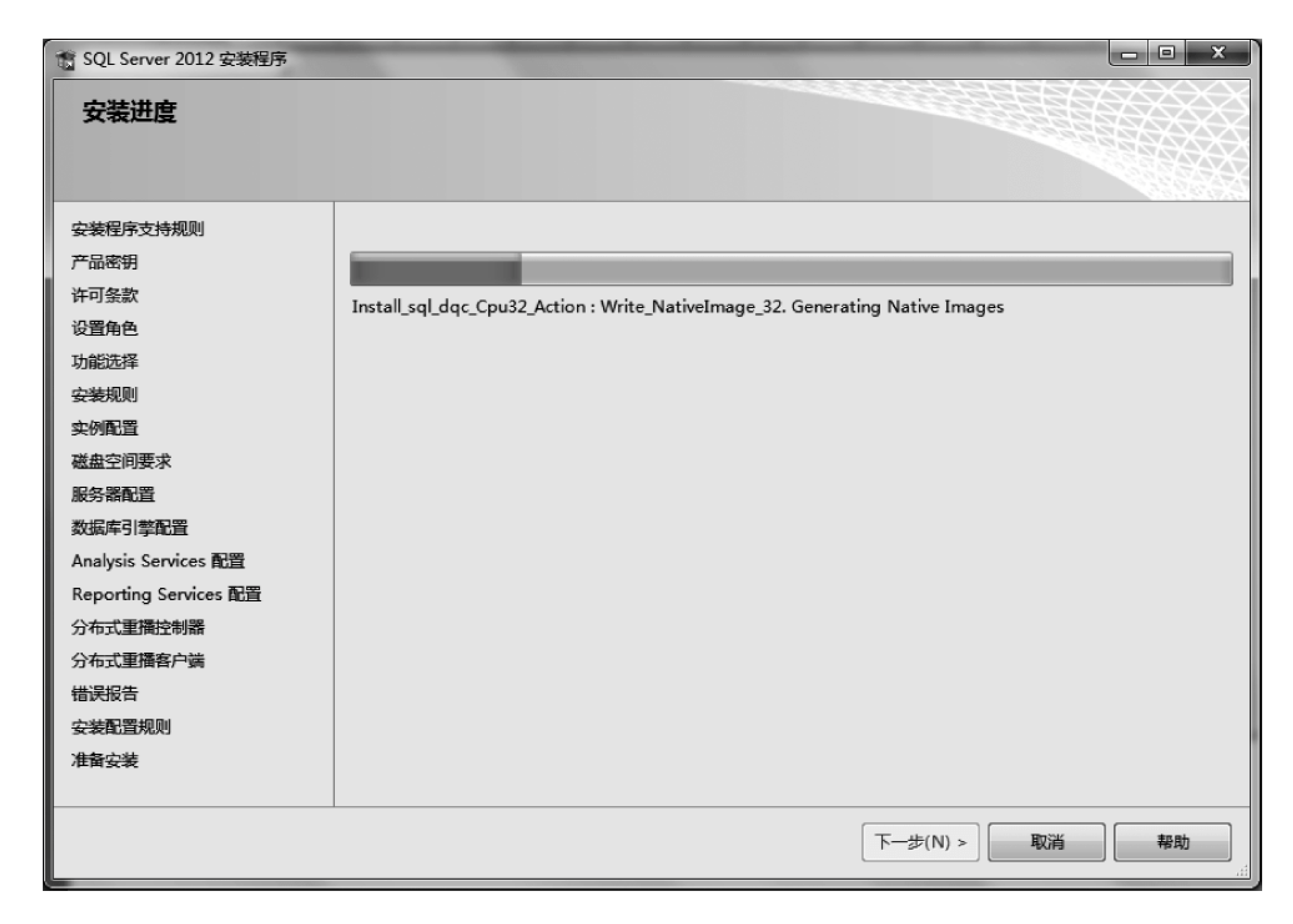


图 2.34 “安装进度”页面

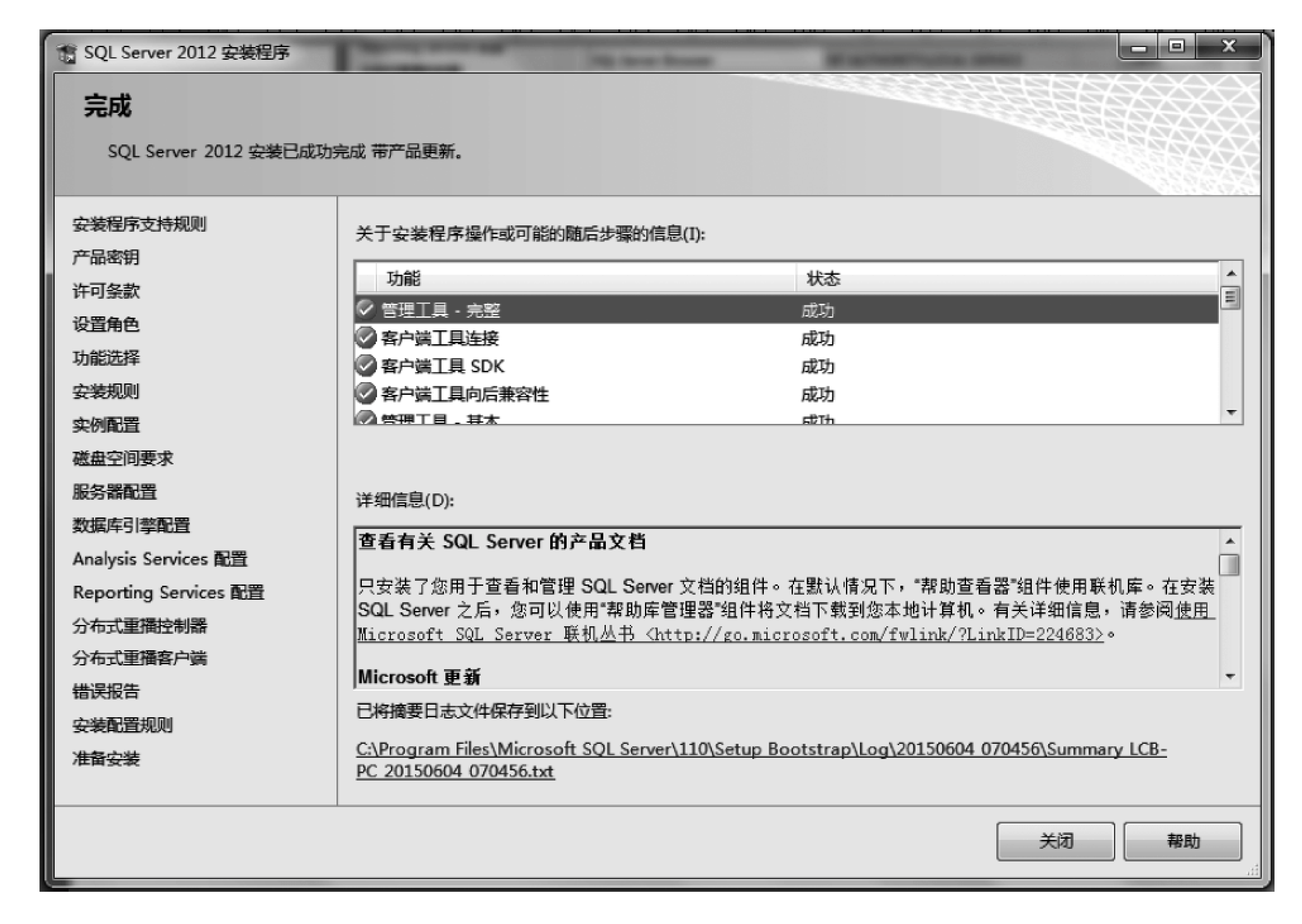


图 2.35 “完成”页面

2.3.2　SQL Server 2012 的主要管理工具

为了满足企业数据管理的需求，SQL Server 提供了不少图形化的管理工具，可以帮助数据库管理员与开发人员更高效地创建、管理和维护 SQL Server 解决方案，从而快速解决复杂的性能与配置问题。

在实际应用中，SQL Server 2012 提供的主要管理工具如表 2.1 所示。

表 2.1　SQL Server 2012 提供的主要管理工具

管理工具	说明
SQL Server Management Studio (SQL Server 管理控制器，SSMS)	用于访问、配置、管理和开发 SQL Server 组件的集成环境，使各种技术水平的开发人员和管理员都能使用 SQL Server
SQL Server 配置管理器	为 SQL Server 服务、服务器协议、客户机协议和客户机别名提供基本配置管理
SQL Server Profiler (SQL Server 事件探查器)	提供了一个图形用户界面，用于监视数据库引擎实例或分析服务实例
数据库引擎优化顾问	可以协助创建索引、索引视图和分区的最佳组合
数据质量客户机	提供了一个简单和直观的图形用户界面，用于连接到 DQS 数据库并执行数据清理操作，还允许集中监视在数据清理操作过程中执行的各项活动
SQL Server 数据工具(SSDT)	提供集成开发环境以便为一些商业智能组件生成解决方案
连接组件	安装用于客户机和服务器之间通信的组件，以及用于 DB-Library、ODBC 和 OLE DB 的网络库

2.3.3　创建数据库 school

在 SQL Server 中使用 SQL Server Management Studio(SQL Server 管理控制器)实现数据库创建和数据表操作。

创建数据库 school 的步骤如下：

① 选择“开始|所有程序|Microsoft SQL Server 2012| SQL Server Management Studio”命令，启动 SQL Server 登录界面，如图 2.36 所示，“服务器类型”选择“数据库引擎”，“服务器名称”设置为 LCB-PC，“身份验证”选择“Windows 身份验证”，单击“连接”按钮。



图 2.36　SQL Server 登录界面

说明：在前面安装 SQL Server 2012 时采用的身份验证模式是“Windows 身份验证”，在安装后用户可以更改身份验证模式。本书中涉及数据库的所有例子均采用“Windows 身份验证”。

② 进入 SQL Server“对象资源管理器”窗口，右击“数据库”节点，在出现的快捷菜单中选择“新建数据库”命令，如图 2.37 所示。



图 2.37　选择“新建数据库”命令

③ 进入“新建数据库”对话框，输入“数据库名称”为 school，单击“数据库文件”列表中每行路径列右侧的 [...] 按钮，更改数据库文件“路径”为“D:\电子商务”目录，如图 2.38 所示，单击“确定”按钮。这样就创建了 school 数据库，其数据库主文件(school.mdf)和日志文件(school_log.ldf)存放在“D:\电子商务”目录中。



图 2.38　创建 school 数据库

2.3.4　在 school 数据库中创建 3 个数据表

在 school 数据库中创建 student 和 score 表的步骤如下：

① 连接到 SQL Server 数据库引擎，在“对象资源管理器”中展开 school 数据库，右击“表”节点，在出现的快捷菜单中选择“新建表”命令，如图 2.39 所示。

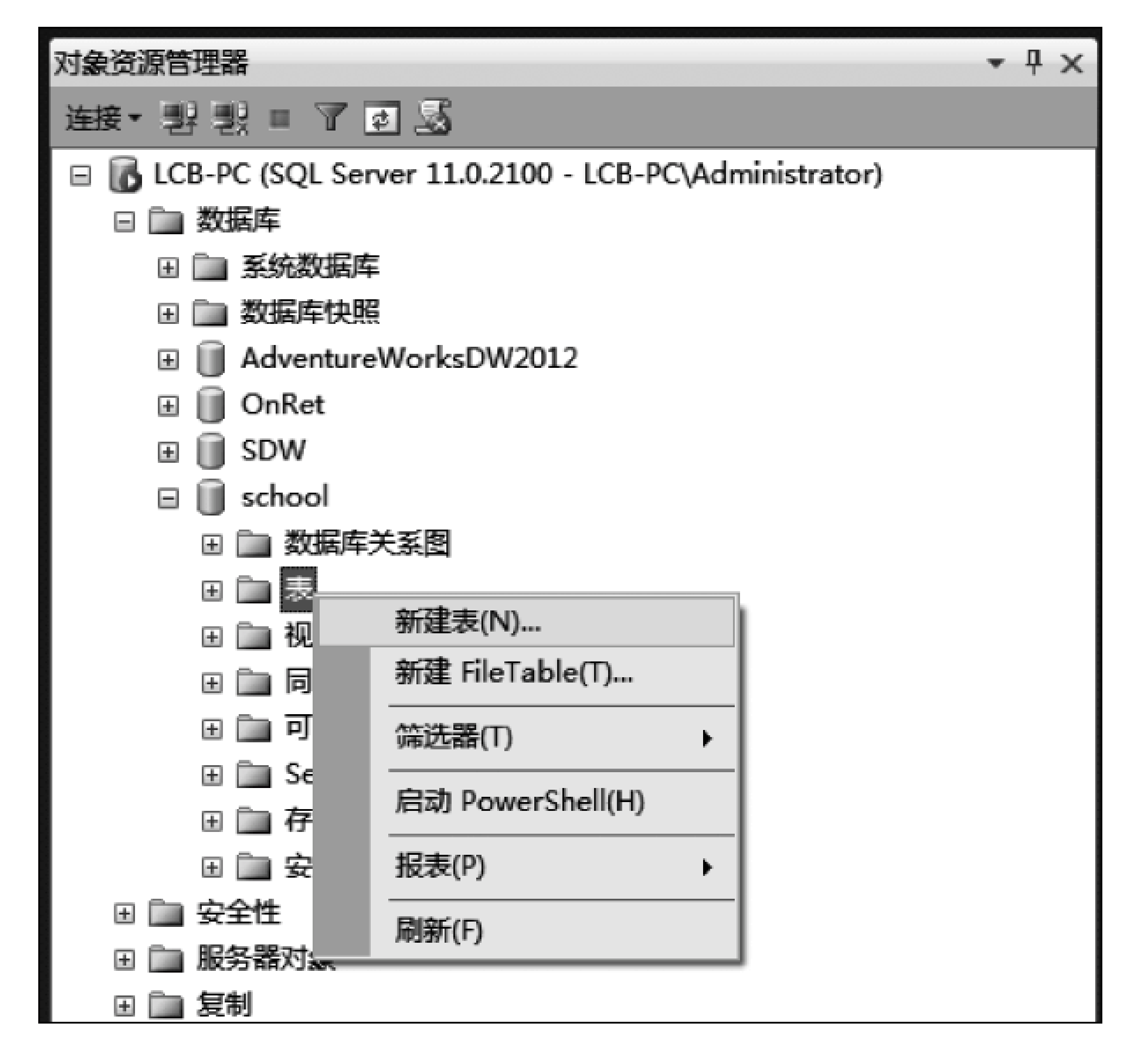
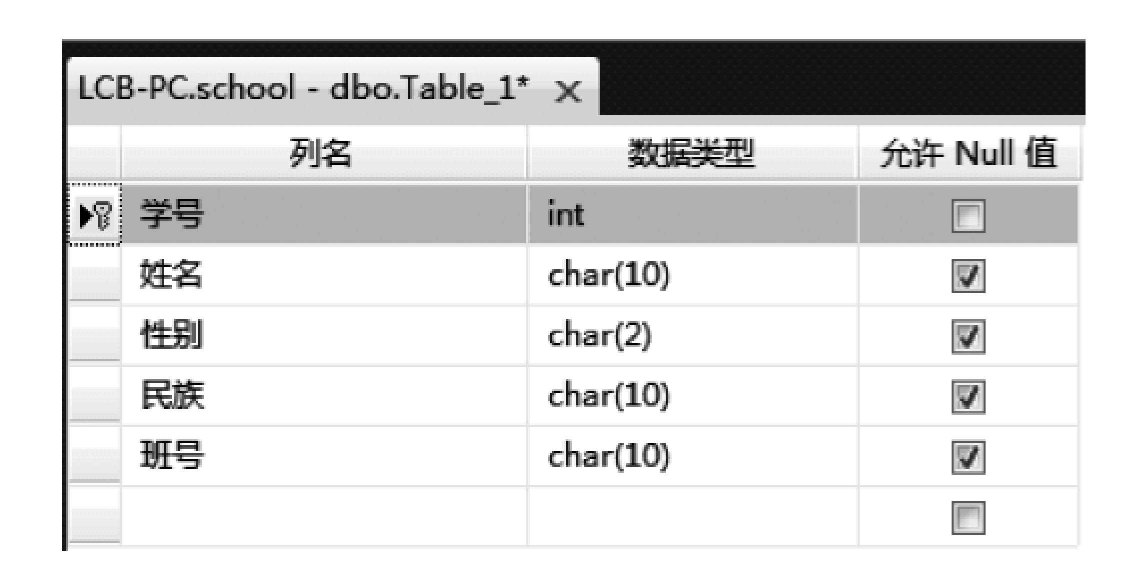


图 2.39　选择“新建表”命令

② 出现创建表结构面板，输入各字段如图 2.40 所示。单击工具栏的 按钮，出现“选择名称”对话框，输入表名称为 student(学生表)，如图 2.41 所示，单击“确定”按钮。



LCB-PC.school - dbo.Table_1*

列名	数据类型	允许 Null 值
学号	int	☐
姓名	char(10)	☑
性别	char(2)	☑
民族	char(10)	☑
班号	char(10)	☑
		☐


图 2.40　创建表结构



图 2.41　“选择名称”对话框

③ 单击工具栏的 新建查询(N) 按钮，在出现的文本编辑框中输入如下 SQL 语句：

```
use school
INSERT INTO student VALUES(1,'王华','女','汉族','15001')
INSERT INTO student VALUES(3,'李兵','男','汉族','15001')
INSERT INTO student VALUES(8,'马棋','男','回族','15002')
INSERT INTO student VALUES(2,'孙丽','女','满族','15002')
INSERT INTO student VALUES(6,'张军','男','汉族','15001')
```

单击工具栏的 ! 执行(X) 按钮，执行上述 SQL 语句，将 5 个学生记录插入到 student 表中。

右击 student 表名，在出现的菜单中选择“选择前 1000 行”命令，结果如图 2.42 所示，表明数据插入成功。

④ 采用相似的操作创建 course 表（课程表），其表结构如图 2.43 所示。

	学号	姓名	性别	民族	班号
1	1	王华	女	汉族	15001
2	2	孙丽	女	满族	15002
3	3	李兵	男	汉族	15001
4	6	张军	男	汉族	15001
5	8	马棋	男	回族	15002

图 2.42 student 表数据

LCB-PC.school -...hool - dbo.course

列名	数据类型	允许 Null 值
课程号	char(10)	☐
课程名	char(20)	☑
		☐

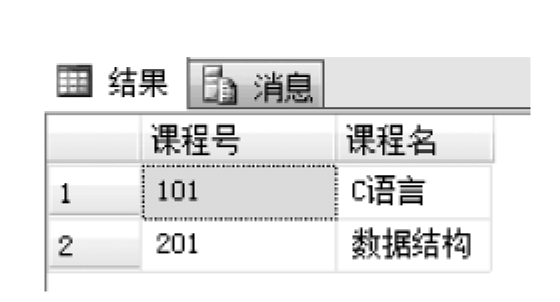
图 2.43 创建 course 表结构

⑤ 输入 course 表记录的 SQL 语句如下：

```
use school
INSERT INTO course VALUES('101','C语言')
INSERT INTO course VALUES('201','数据结构')
```

插入成功后 course 表中数据如图 2.44 所示。

⑥ 采用相似的操作创建 score 表，其表结构如图 2.45 所示。



	课程号	课程名
1	101	C语言
2	201	数据结构

图 2.44 course 表数据



LCB-PC.school - dbo.score

列名	数据类型	允许 Null 值
学号	int	☐
课程号	char(10)	☐
分数	float	☑
		☐

图 2.45 创建 score 表结构

⑦ 输入 score 表记录的 SQL 语句如下：

```
use school
INSERT INTO score VALUES(1,'101',80)
INSERT INTO score VALUES(3,'101',76)
INSERT INTO score VALUES(8,'101',88)
INSERT INTO score VALUES(2,'101',70)
INSERT INTO score VALUES(6,'101',90)
INSERT INTO score VALUES(1,'201',83)
INSERT INTO score VALUES(3,'201',70)
INSERT INTO score VALUES(8,'201',79)
INSERT INTO score VALUES(2,'201',52)
INSERT INTO score VALUES(6,'201',92)
```

插入成功后 score 表中数据如图 2.46 所示。

说明：这里创建的 school 数据库及包含的 3 个数据表，将作为第 10 和第 11 章介绍 ADO .NET 编程的样本数据。

	学号	课程号	分数
1	1	101	80
2	1	201	83
3	2	101	70
4	2	201	52
5	3	101	76
6	3	201	70
7	6	101	90
8	6	201	92
9	8	101	88
10	8	201	79

图 2.46 score 表数据

2.4 练 习 题

1. 单项选择题

(1) IIS 用于创建、管理和承载 ASP .NET 网站的 Web 服务器,它的英文全称是( )。

A. Internet Information Services B. Internet Information System

C. Inter Information Services D. Inter Information System

(2) Visual Studio 2012 支持多种计算机语言开发,所支持的语言称为兼容语言。兼容语言不包括( )。

A. C# B. VB C. Pascal D. C++

(3) VisualStudio2012 不适合开发( )程序。

A. Web 应用程序 B. 3D 动画

C. Web 服务 D. Windows 窗体应用程序

(4) 以下叙述中错误的是( )。

A. ASP .NET 提供了多种语言支持

B. ASP .NET 可以开发多种类型的应用程序

C. ASP .NET 提供跨平台支持,也可以在 UNIX 下执行

D. ASP .NET 采取编译执行的方式,极大地提高了运行的性能

(5) SQL Server 2012 是一个( )。

A. Web 开发工具 B. 数据库管理系统

C. C#语言编译环境 D. 网页制作工具

(6) 在安装 SQL Server 2012 时,可以指定两种身份验证模式,它们是( )。

A. Windows 身份验证模式和防火墙

B. 混合身份验证模式和防火墙

C. SQL Server 身份验证和防火墙

D. Windows 身份验证模式和混合模式

(7) 在 SQL Server 中使用( )管理工具创建数据库。

A. SQL Server 分析服务 B. SQL Server 管理控制器

C. SQL Server 报表服务 D. SQL Server 数据工具

(8) 以下叙述中错误的是( )。

A. 一台计算机安装了 IIS 后能够响应网页请求,它就是一台 Web 服务器

B. 一台计算机安装了 SQL Server 后能够响应 SQL 请求,它就是一台数据库服务器

C. 一台计算机可以作为 Web 服务器,也可以作为数据库服务器

D. 一台计算机要么作为 Web 服务器,要么作为数据库服务器,不能同时作为 Web 服务器和数据库服务器

(9) 小王正在家里通过拨号上网访问搜狐网站,以下叙述正确的是( )。

A. 小王的机器是服务器端,搜狐网站是客户端

B. 搜狐网站是服务器端,小王的机器是客户端

C. 小王的机器既是服务器端,又是客户端

D. 以上说法全不对

(10) 小王正在家里计算机上开发和调试ASP.NET网站,以下叙述正确的是(　　)。

A. 小王的计算机是服务器端,但不是客户端

B. 小王的计算机是客户端,但不是服务器端

C. 小王的计算机既是服务器端,又是客户端

D. 以上说法全不对

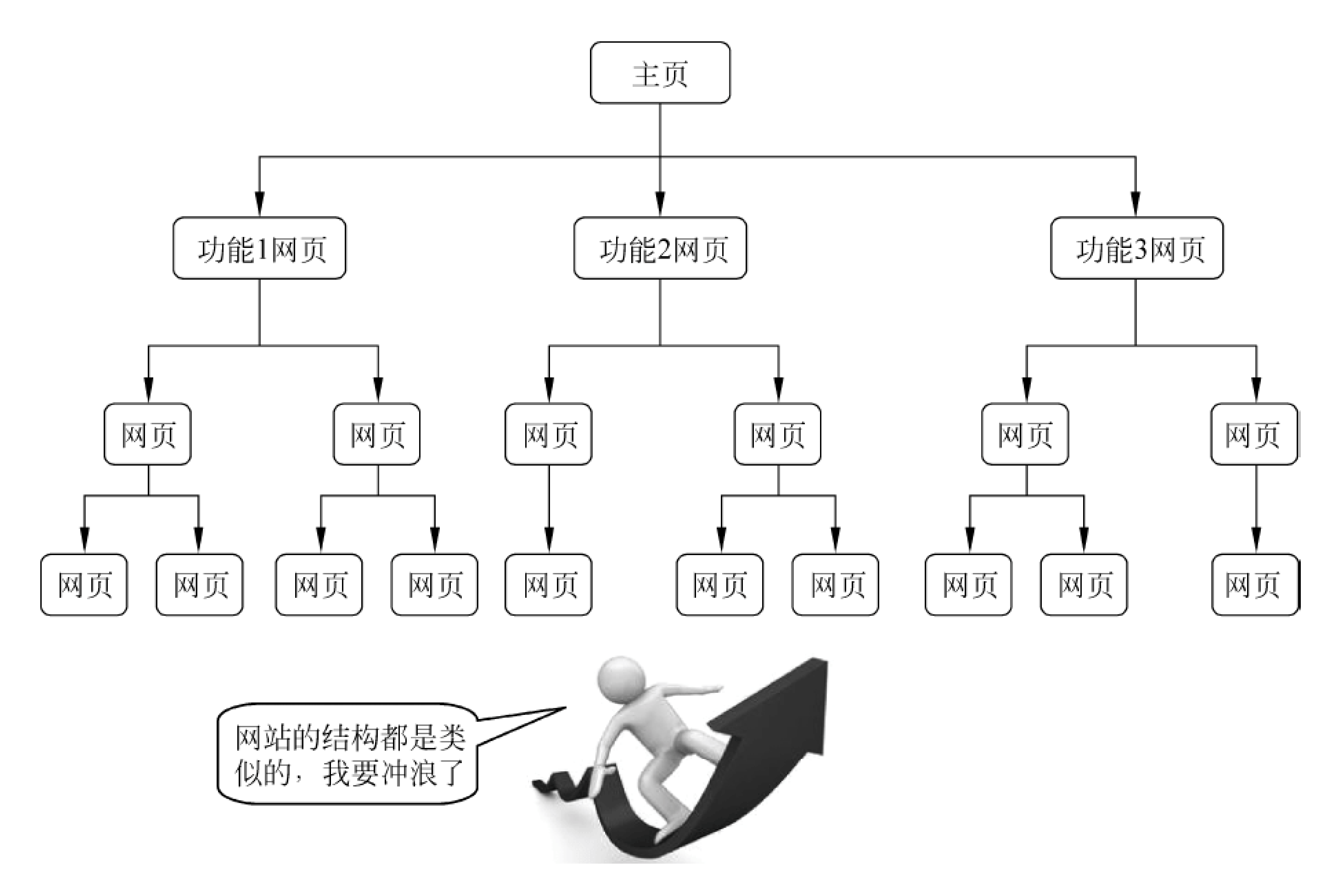
2. 问答题

(1) 简述Visual Studio 2012的安装步骤。

(2) 简述SQL Server 2012的安装步骤。

(3) 在用Visual Studio 2012开发Web应用程序时,为什么要安装.NET Framework?

第 3 章　ASP .NET 网站结构

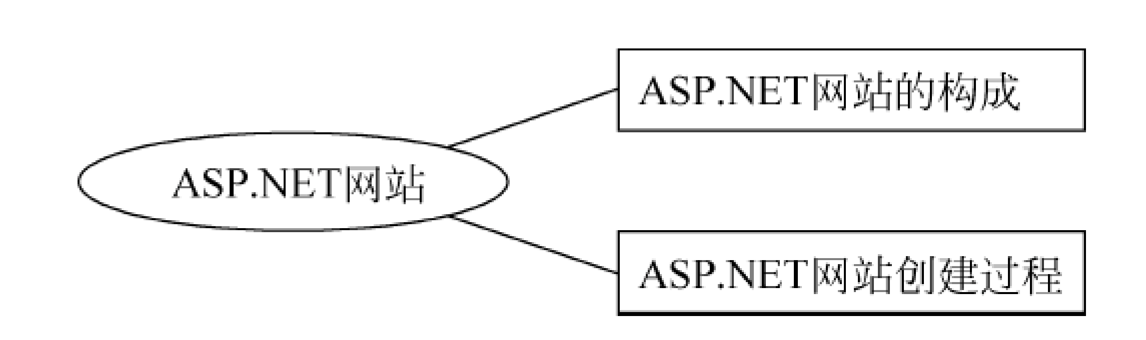


本章指南



3.1　ASP .NET 网站

知识梳理



3.1.1 ASP.NET网站的构成

图3.1展示了一个ASP.NET网站OnRetS的结构。从中看出一个典型的ASP.NET网站通常由多个网页文件组成,每个网页将共享许多通用的资源和配置设置。通常网页文件和相关的资源文件按内容存放在不同的目录中。

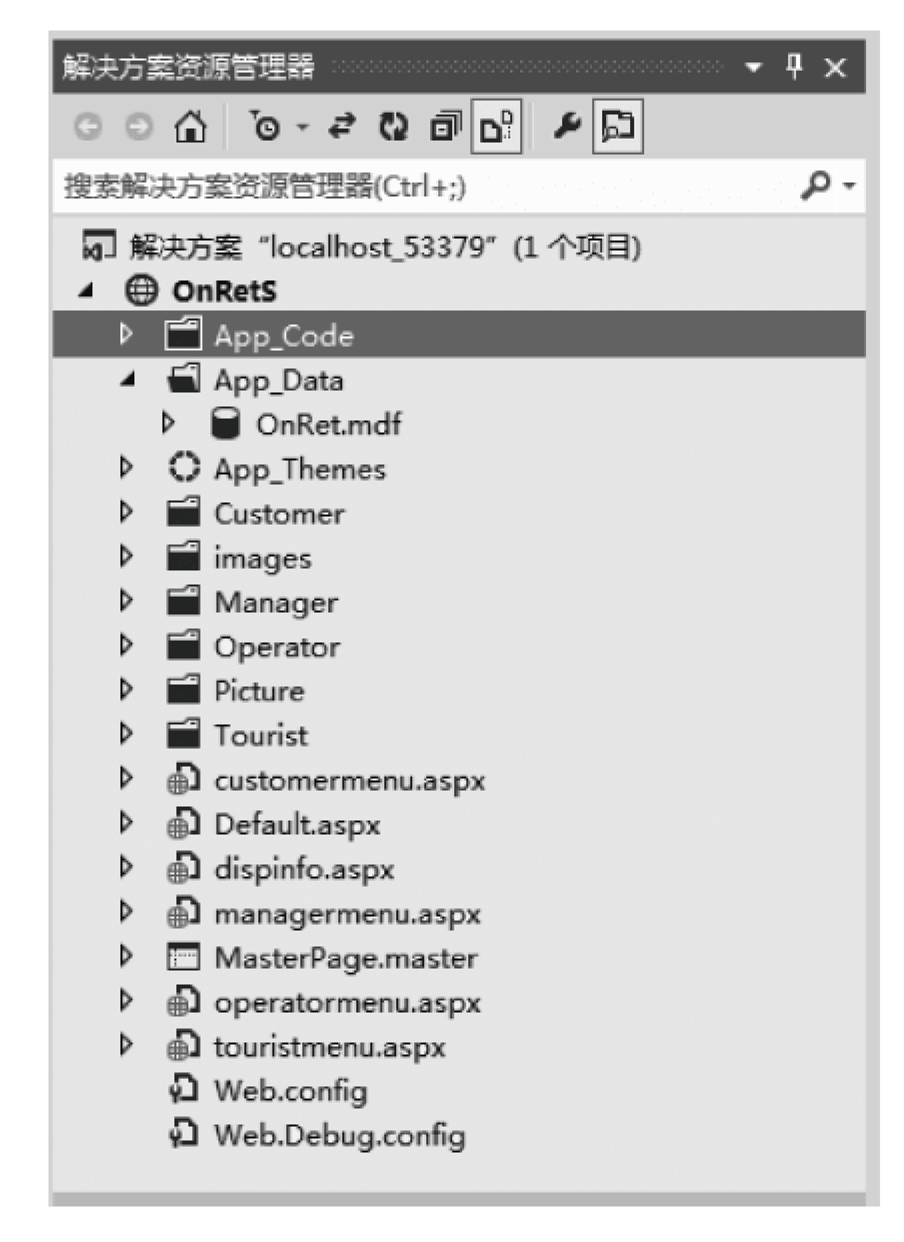


图3.1 OnRets网站的结构

1. 常见的文件类型

从文件组成看,网站由多个文件组成,包括Web网页、配置文件和数据库文件等。常见的文件类型如下。

- *.aspx: ASP.NET网页文件,包含了用户界面的代码,如图3.1中Default.aspx是OnRetS网站的主页。
- *.cs: 采用代码隐藏页模型设计网页时的代码隐藏文件,如选择C#作为开发语言就产生.cs文件。
- *ascx: 开发人员自己设计的ASP.NET用户控件。
- web.config: ASP.NET应用程序的基于XML格式的配置文件,包含各种ASP.NET功能的配置信息,如数据库连接、安全设置、状态管理等。Web.Debug.config是调试模式下的web.config文件。
- global.asax: 全局应用程序文件,该文件驻留在ASP.NET网站的根目录下。
- *.asmx文件: 为其他计算机提供共享应用程序的Web服务。
- *.skin文件: 外观文件,用于设置主题。
- *.css文件: 样式表文件。
- *.master: 母版页文件。例如,图3.1中MasterPage.master就是一个母版页文件,用作Default.aspx等网页的母版页。

2. 常用的目录

一个网站包含若干目录。为了更方便管理和使用,ASP.NET保留了一些可用于特定内容的文件和目录名称。一个网站常用的目录如下。

- App_Data目录: 用于存放应用程序使用的数据库。它是一个集中存储应用程序所用数据库的地方。App_Data目录可以包含SQL Server数据库文件(.mdf)、Access数据库文件(.mdb)或XML文件等。例如,在图3.1中,App_Data目录中存放OnRet数据库文件。
- App_Code目录: 用于存放所有应当作为应用程序的一部分动态编译的类文件。在开发网站时,对App_Code目录的更改会导致整个应用程序的重新编译。例如,在图3.1中,App_Code目录存放网站的一些通用C#代码。
- Bin目录: 包含应用程序所需的,用于控件、组件或需要引用的任何其他代码的可部署程序集。该目录中存在的任何.dll文件将自动地链接到应用程序。

- App_Themes 目录：用于存放主题文件和 CSS 文件等。例如，在图 3.1 中，App_Themes 目录存放 Blue 主题和 StyleSheet.css 文件。
- App_GlobalResources 目录：是全局资源文件目录，可以存放一些资源文件，这些资源文件中包含指向图像或其他数据的资源字符串，在网站的所有网页中都可以访问这些资源字符串。
- 其他目录：网站开发人员可以根据需要自定义目录。例如，在图 3.1 中，images 和 Picture 目录用于存放图片文件，Operator 用于存放操作员网页文件等。

3.1.2　ASP .NET 网站创建过程

下面通过一个例子说明创建 ASP .NET 网站的过程。

【练一练】 创建一个文件系统网站 CH3 的操作步骤如下：

① 启动 Visual Studio 2012。

② 选择“文件|新建|网站”命令，出现“新建网站”对话框，单击模板列表中 Visual C# 项，列出已安装的网站模板。各网站模板的说明如下。

- ASP .NET 空网站：该模板只包含一个配置文件(web.config)。本书中主要使用该模板构建网站示例，并逐步添加文件和目录。
- ASP .NET 窗体网站：该模板用于配置一个基本的 ASP .NET 网站，包含许多文件和目录用于开始网站的开发。
- ASP .NET 网站(Razor v1 或 Razor v2)：通过微软的 Web Pages 框架，使用这些模板来创建网站。
- ASP .NET Dynamic Data 实体网站：该模板用于创建灵活且强大的 Web 网站来管理数据库中的数据，而不需要手工输入许多代码。
- WCF 服务：该模板用于创建包含一个或多个 WCF 服务的网站。
- ASP .NET 报表网站：该模板用于创建企业报表网站。

这里选择“ASP .NET 空网站”模板。

③ “Web 位置”选项有如下 3 种。

- 文件系统：如果主机没有安装 IIS，也不想设置服务器的位置等信息，可以使用这个设置，表示代码源文件放在选定本地文件目录中。Visual Studio 会把用户指定的路径视为该网站的根目录，并在预览时启动内置的网页服务器，根据这个位置来模拟执行，并可以很方便地进行程序源代码移植。
- HTTP：如果主机已经安装了 IIS，便可以使用这个设置。这个设置与 IIS 的设置相关，还必须设置网页服务器的预览网址，所设计的文件也会放置在 IIS 所设置网站的根目录中。在根目录下还可以设置若干虚拟目录。
- FTP：如果测试主机并不在本机上，可以使用这个设置，表示将代码源文件保存在远程的 FTP 服务器上。Visual Studio 通过文件传输协议 FTP 访问网站，这样更容易访问其他服务器上的网站。在第一次运行 FTP 网站时提示用户指定服务器 URL。

这里选择“Web 位置”为“文件系统”。

④ 单击“浏览”按钮，选择“D:\电子商务\CH3”目录，如图 3.2 所示，单击“确定”按钮。这样就创建了一个空网站 CH3，其中只有一个配置文件 web.config。

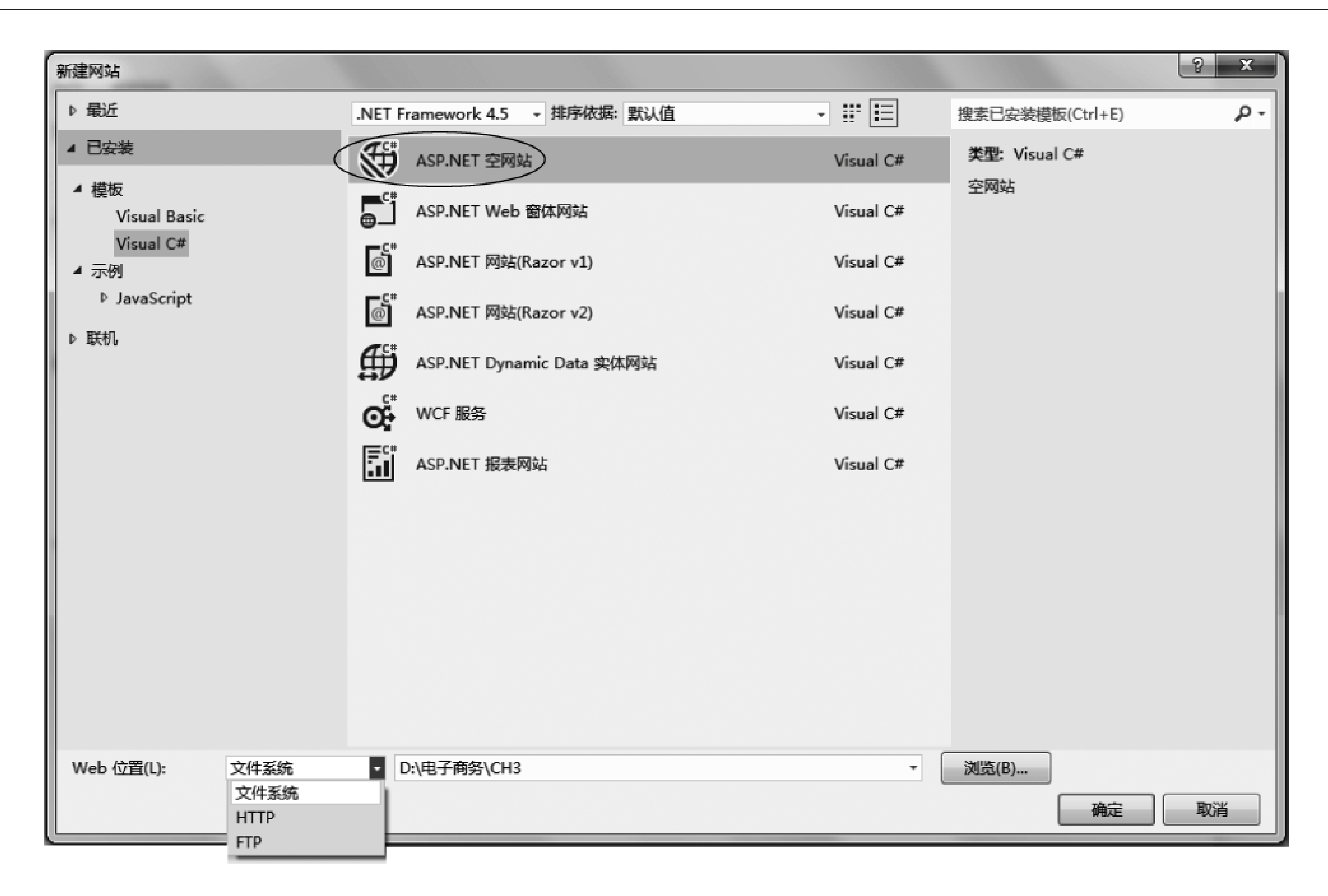
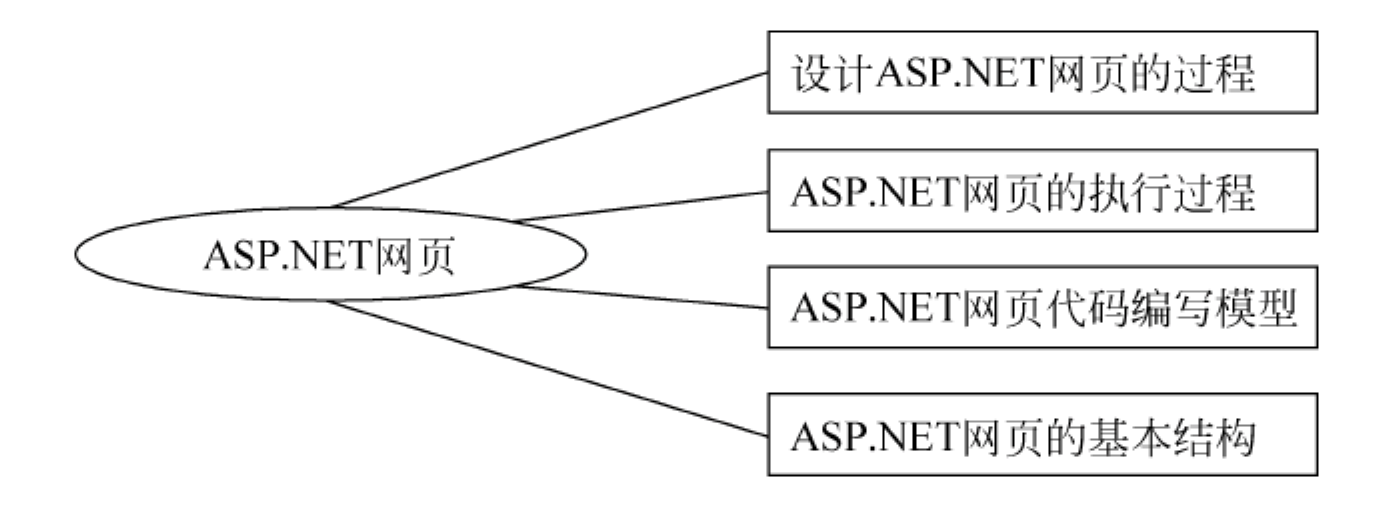


图 3.2 网站的模板

3.2 ASP .NET 网页

知识梳理



3.2.1 设计 ASP .NET 网页的过程

下面通过一个例子说明创建 ASP .NET 网页的过程。

【练一练】 在前面创建的 CH3 网站中设计一个 webform1 网页，其功能类似于 1.2.5 节介绍的静态网页 spage.html。

其设计步骤如下：

① 启动 Visual Studio 2012。

② 选择"文件|打开|网站"命令，出现如图 3.3 所示的"打开网站"对话框。

其中，左边的列表表示打开网站的类型。

- 文件系统：从磁盘位置打开网站。
- 本地 IIS：使用本地计算机上的 IIS 打开网站。



图 3.3 "打开网站"对话框

- FTP 站点：从 FTP 位置打开网站。
- 远程站点：打开 FrontPage 网站。
- 源代码管理：从源代码管理打开网站。

这里选择"文件系统"，并单击"D:\电子商务\CH3"文件，单击"打开"按钮。

③ 出现如图 3.4 所示的 Microsoft Visual Studio 对话框提示是否使用 IIS Express。因为在 Visual Studio 中开发网站时，需要 Web 服务器才能测试或运行它们。Visual Studio 2010 之前的版本内置了 Visual Studio Development Server（Visual Studio 开发服务器）用于测试或运行网站，Visual Studio 2010 版本开始有了 IIS Express。在 Visual Studio 2012 中可以使用 Visual Studio 开发服务器和 IIS Express 测试或运行网站，最好使用 IIS Express，因为



图 3.4 Microsoft Visual Studio 对话框

微软公司表示以后的 Visual Studio 版本可能不再支持 Visual Studio 开发服务器。

单击“是”按钮，表示使用 IIS Express。此时就打开 CH3 网站。

④ 在 CH3 网站中，选择“网站|添加新项”命令，出现“添加新项”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform1. aspx，其他保持默认项，如图 3. 5 所示，单击“添加”按钮，这样在 CH3 网站中就添加了一个名称为 webform1. aspx 的空网页。



图 3.5 “添加新项”对话框

⑤ 在系统集成界面的中间部分出现 webform1. aspx 网页的源视图代码，单击下方的“设计”选项卡，进入网页的可视化设计界面，从工具箱中拖曳两个标签到网页中，名称分别为 Label1 和 Label2，将它们的 Text 属性分别修改为“-电子商务网站-”和“欢迎光临”，再通过属性窗口修改它们的字体，如图 3. 6 所示。

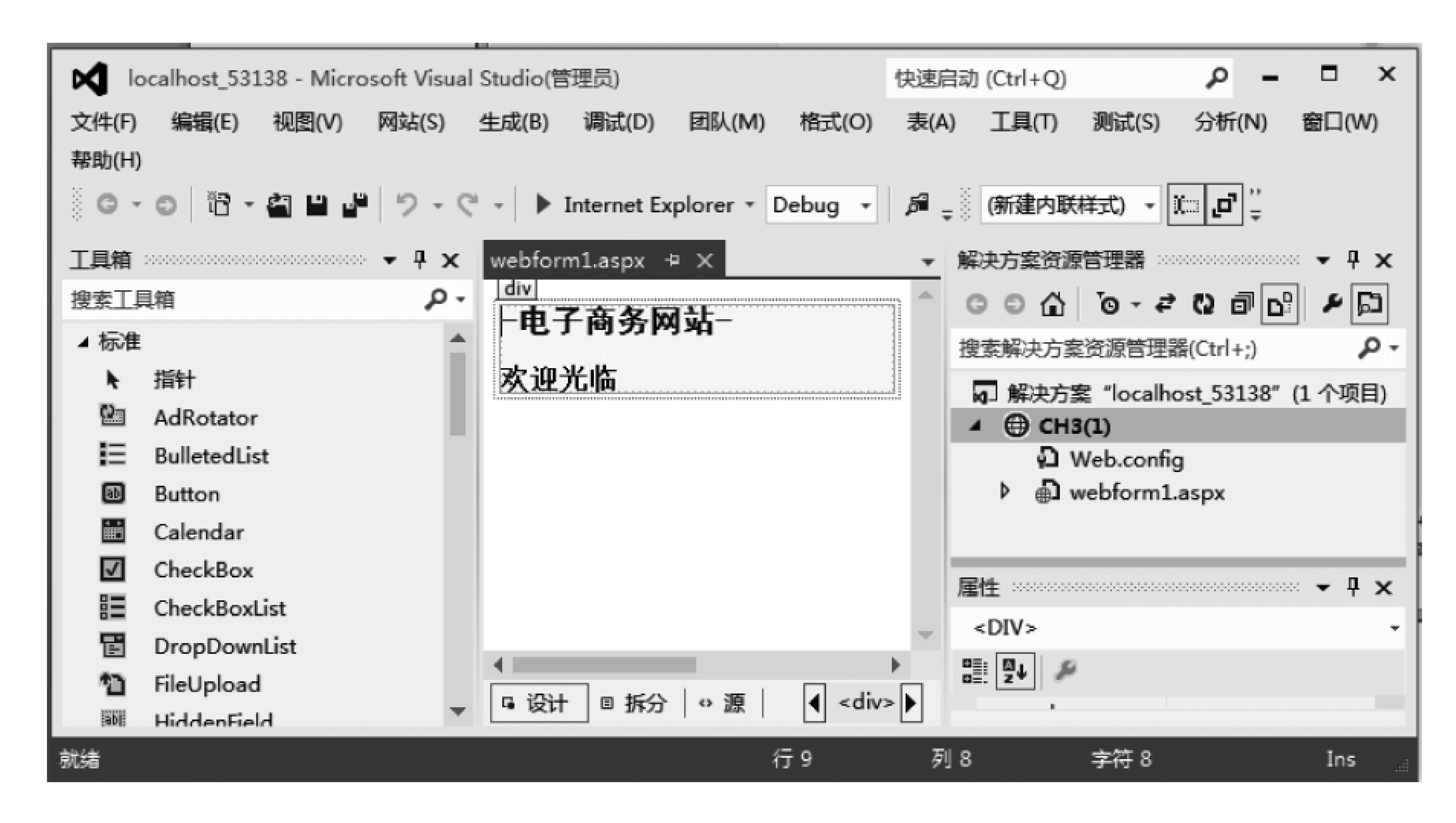


图 3.6 “添加新项”后的显示

说明：网页源视图代码编辑窗口下方有 3 个选项卡，其中 设计 选项卡用于可视化设计网页；源 选项卡用于显示网页的源视图代码；拆分 选项卡用于显示网页元素与对应的源视图代码关联。

⑥ 单击 源 选项卡，看到 webform1 网页的源视图代码(称为 ASP .NET 网页代码)如下：

```
<%@ Page Language = "C#" AutoEventWireup = "true"
    CodeFile = "webform1.aspx.cs" Inherits = "webform1" %>
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
   <head runat = "server">
      <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8"/>
      <title>webform1 网页</title>
   </head>
   <body>
     <form id = "form1" runat = "server">
       <div>
        <asp:Label ID = "Label1" runat = "server" Text = "-电子商务网站-"
            style = "font-size: large; font-weight: 700; font-family: 黑体">
        </asp:Label>
        <br /><br />
        <asp:Label ID = "Label2" runat = "server" Text = "欢迎光临"
            style = "font-size: medium; font-weight: 700">
        </asp:Label>
       </div>
    </form>
   </body>
</html>
```

从中看到，Visual Studio 将开发人员的可视化设计操作转换为相应的网页代码。例如，在网页中拖曳一个标签 Label1 并设置其属性的代码如下：

```
<asp:Label ID = "Label1" runat = "server" Text = "-电子商务网站-"
    style = "font-size: large; font-weight: 700; font-family: 黑体"></asp:Label>
```

其中，“asp:”是 ASP .NET 控件标识符；ID 属性指定标识号(网页中每个元素的标识号是唯一的)；style 定义该标签在浏览器中的显示样式，包括字体和字体大小等。

⑦ 单击工具栏的 ▶ Internet Explorer (将 IE 浏览器设置为默认的浏览器)浏览该网页。在第一次浏览时，系统会询问是否要在<Web. config>文件中添加调试功能，如图 3.7 所示，单击“确定”按钮即可。在以后浏览时就不再出现询问对话框。如果按 Ctrl+F5 键，表示在非调试状态浏览网页，会立即出现 webform1 网页的浏览界面。webform1 网页执行界面如图 3.8 所示。



图 3.7　“未启用调试”对话框

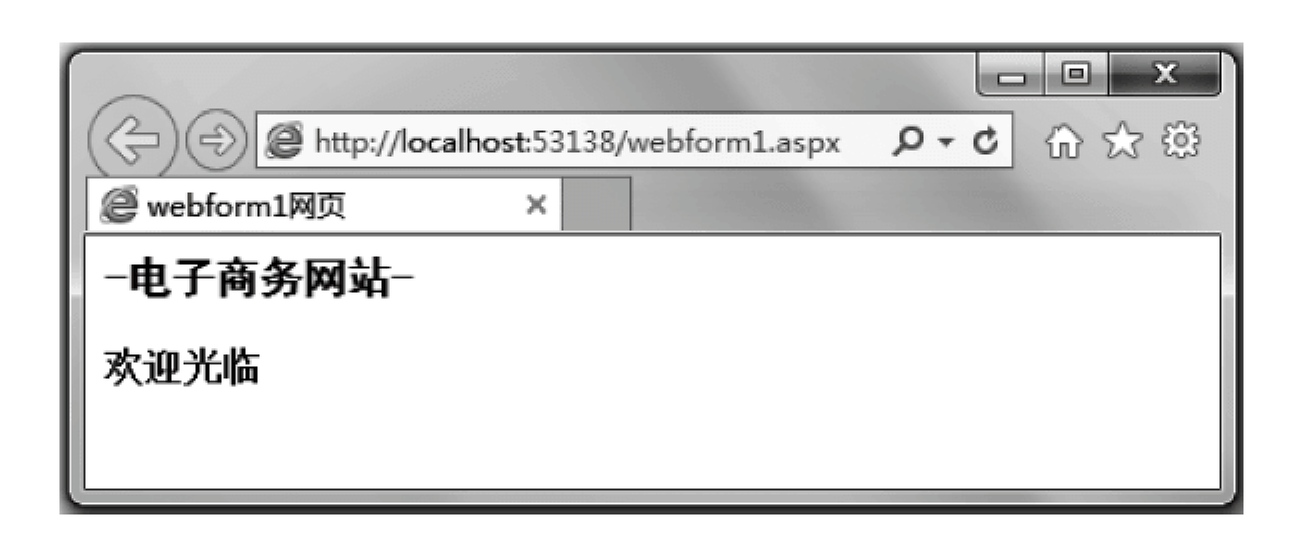


图 3.8 webform1 网页执行界面

3.2.2 ASP .NET 网页的执行过程

在图 3.8 的浏览界面中右击，在出现的快捷菜单中选择"查看源"命令，显示的纯 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head><meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>webform1 网页</title>
    </head>
    <body>
        <form method="post" action="webform1.aspx" id="form1">
          <div class="aspNetHidden">
               <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
                  value="0XDZ3/n/lKuptIJQ8PIKqKZR1iC40XFJWTjE/jCLRI91ZJ
                  u3WyJ+jV+ZaEXUE2xpmDiqEliCNQmH8Vq4jcXRpPwjmW4oJxqb/e0HwxuyDuA=" />
         </div>
         <div class="aspNetHidden">
            <input type="hidden" name="__VIEWSTATEGENERATOR"
               id="__VIEWSTATEGENERATOR" value="C687F31A" />
         </div>
         <div>
            <span id="Label1" style="font-size: large; font-weight: 700; font-family:
               黑体">-电子商务网站-</span>
            <br />
            <br />
            <span id="Label2" style="font-size: medium; font-weight: 700">
               欢迎光临</span>
         </div>
       </form>
    </body>
</html>
```

上述代码是纯 HTML。其中有两个隐藏域，name 为_VIEWSTATE 的隐藏域，只有 form 表单(窗体)标识 runat 为 server 时才会自动生成。当请求某个网页时，ASP .NET 把所有控件的状态序列化成一个字符串，然后作为网页的隐藏属性送到客户端。当客户端把网页回传时，ASP .NET 分析回传的网页窗体属性，并赋给控件对应的值。name 为_VIEWSTATEGENERATOR 的隐藏域，用于在 ASP .NET 运行时帮助确定是回传到相同的网页还是跨页。

从中看到，ASP .NET 网页代码与浏览器中最终显示的纯 HTML 代码是不同的，这种转换是由 ASP .NET 引擎完成的。实际上，像 IE 这样的浏览器并不认识 ASP .NET 网页代码。

一个典型的 ASP .NET 网页执行过程如图 3.9 所示。图中的各个步骤说明如下：

① 客户机通过浏览器发出 Web 请求(请求访问某个 ASP .NET 网页)。

② Web 服务器收到 ASP .NET 动态网页请求。

③ Web 服务器从硬盘的指定位置查找相应的 ASP .NET 动态网页文件。

④ 将硬盘中找到的 ASP .NET 动态网页文件返回给 Web 服务器。

⑤ Web 服务器将 ASP .NET 动态网页发给 ASP .NET 引擎。

⑥ ASP .NET 引擎会逐行地读取该文件,并执行文件中的程序代码(脚本),如果需要访问数据库,则将这部分代码发给数据库服务器;如果不需要访问数据库,则直接转到步骤⑧。

⑦ 数据库服务器执行数据库访问,并将结果返回给 ASP .NET 引擎。

⑧ ASP .NET 引擎生成最终的纯 HTML 文件并返回给 Web 服务器。

⑨ Web 服务器将该纯 HTML 文件发送给客户机。

⑩ 客户机收到请求的纯 HTML 文件,并在浏览器中以图形方式将 HTML 标记显示在计算机屏幕上。

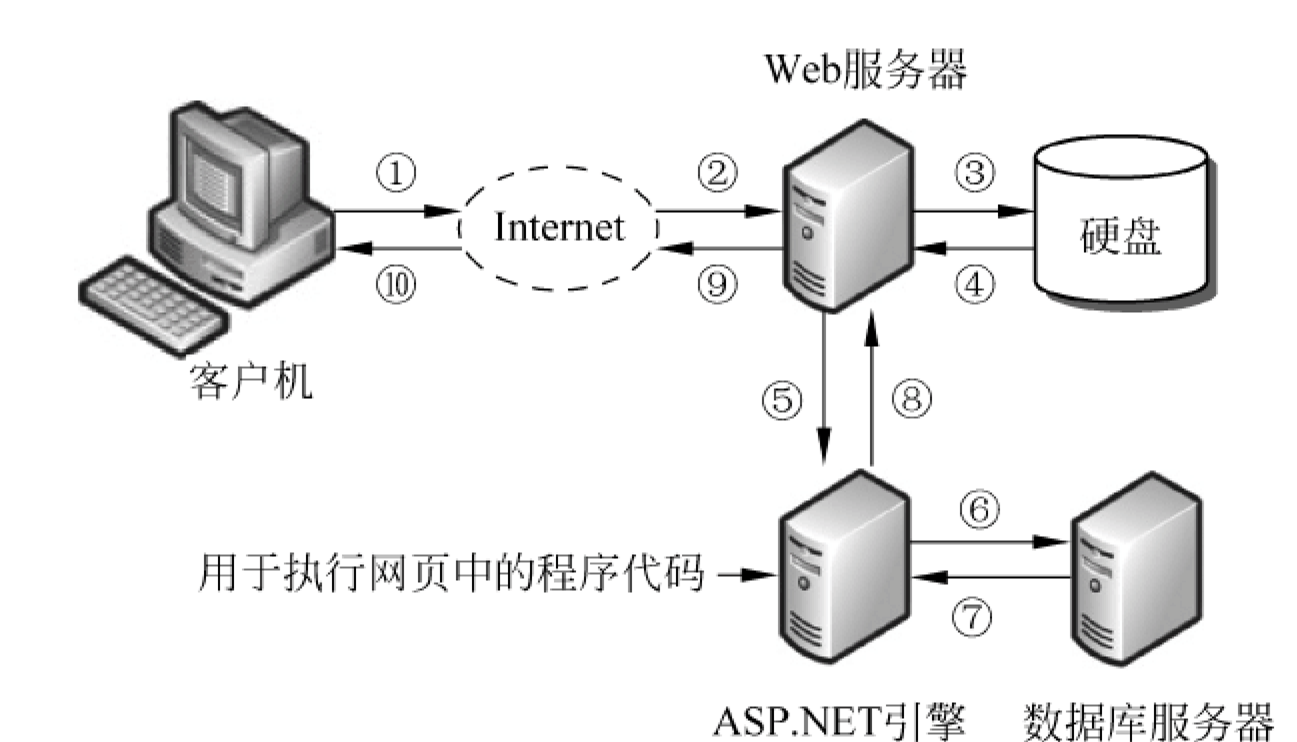


图 3.9　ASP .NET 网页的执行由 ASP .NET 引擎处理

因此,ASP .NET 网页是在服务器上执行的,通过 ASP .NET 引擎处理将其转换为纯 HTML 代码,然后由浏览器执行。图 3.10 所示是 Label1 控件的转换过程。

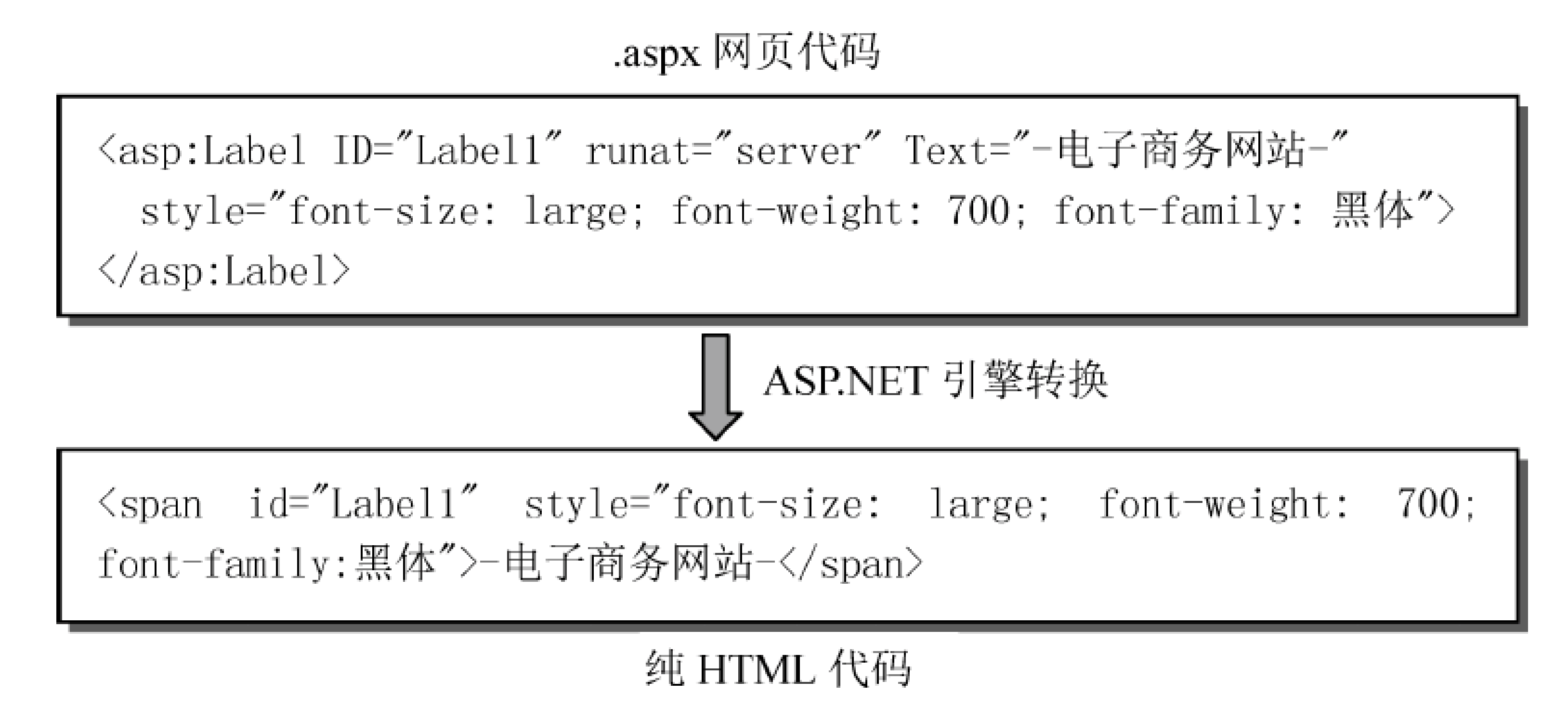


图 3.10　ASP .NET 进行的代码转换

3.2.3　ASP .NET 网页代码编写模型

在 ASP .NET 网页中,开发人员的编程工作分为两个部分:可视元素和编程逻辑。可视元素部分包括标记、服务器控件和静态文本的文件组成。编程逻辑部分包括事件处理程序和其他代码,这些代码由用户创建与网页进行交互。程序代码可以驻留在网页的 script 块或单

独的类中。如果代码在单独的类文件中，则该文件称为“代码隐藏”文件。在本书中编程逻辑代码使用 C#语言编写。

因此，ASP .NET 提供两种代码编写模型：单文件页模型和代码隐藏页模型。这两个模型功能相同，在两种模型中可以使用相同的控件和代码。

1. 单文件页模型

在单文件页模型中，网页的标记及其程序代码位于同一个. aspx 文件中，也称为内联模型。程序代码位于<script>元素中，它包含 runat＝"server"属性，此属性将其标记为 ASP .NET 引擎应执行的代码。

【练一练】 在 CH3 网站中添加一个单文件页模型的网页 webform2，其功能是实现用户输入的两个整数相加运算。设计步骤如下：

① 进入 CH3 网站，选择“网站|添加新项”命令，出现的“添加新项”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform2. aspx，去掉“将代码放在单独的文件中”的勾选，如图 3.11 所示，单击“添加”按钮，这样在 CH3 网站中添加一个名称为 webform2. aspx 的单文件页模型网页。



图 3.11　添加 webform2 网页

② 单击 设计 选项卡，进入网页的可视化设计界面，从工具箱中拖曳 3 个标签 Label1～Label3、两个文本框 TextBox1 和 TextBox2、一个命令按钮 Button1 到设计界面中。通过属性窗口将 Label1 和 Label2 的 Text 属性分别更改为“整数 1”和“整数 2”。按下 Ctrl 键，选择 Label1 和 Label2，选择“格式|A 字体”命令，设计其字体如图 3.12 所示。采用同样的操作设置其他控件的 Text 属性和字体。最终的设计界面如图 3.13 所示。

③ 双击“相加”按钮，在<script runat＝"server">和</script>之间添加如下代码：

```
protected void Button1_Click(object sender, EventArgs e)
{   int a = int.Parse(TextBox1.Text.Trim());
    int b = int.Parse(TextBox2.Text.Trim());
    int c = a + b;
    Label3.Text = "两个整数相加结果：" + c.ToString();
```

```
}
```



图 3.12　“字体”对话框



图 3.13　webform2 网页设计界面

④ 单击 源 选项卡，看到 webform2 网页的源视图代码如下：

```
<%@ Page Language="C#" %>
<!DOCTYPE html>
<script runat="server">
    protected void Button1_Click(object sender, EventArgs e)
    {   int a = int.Parse(TextBox1.Text.Trim());
        int b = int.Parse(TextBox2.Text.Trim());
        int c = a + b;
        Label3.Text = "两个整数相加结果: " + c.ToString();
    }
</script>
<html xmlns="http://www.w3.org/1999/xhtml">
   <head runat="server">
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
      <title></title>
      <style type="text/css">
         .auto-style1 {
            font-family: 楷体;
            font-weight: bold;
            font-size: medium;
            color: #0000FF;
         }
      </style>
   </head>
   <body>
      <form id="form1" runat="server">
        <div>
          <asp:Label ID="Label1" runat="server" Text="整数 1"
              CssClass="auto-style1"></asp:Label>
          <asp:TextBox ID="TextBox1" runat="server" Width="74px"></asp:TextBox>
            
```

```
          <asp:Label ID="Label2" runat="server" Text="整数 2"
               CssClass="auto-style1"></asp:Label>
          <asp:TextBox ID="TextBox2" runat="server" Width="80px"></asp:TextBox>
        </div>
      <div>
        <br />
                 
        <asp:Button ID="Button1" runat="server" Text="相加" OnClick="Button1_Click"
           style="color: #FF0000; font-size: medium; font-weight: 700;
                font-family: 黑体" />
      </div>
      <div>
        <br />
        <asp:Label ID="Label3" runat="server"
            style="color: #FF00FF; font-size: medium; font-weight: 700;
                 font-family: 仿宋"></asp:Label>
      </div>
    </form>
  </body>
</html>
```

说明：<style>元素中的.auto-style1 样式是由 Visual Studio 自动生成的，将开发人员的可视化设计操作转换为样式并自动通过 CssClass 属性作用于相关的可视元素。

⑤ 单击工具栏的 ▶ Internet Explorer ▾ 浏览该网页。在两个文本框中分别输入 8 和 15，如图 3.14 所示，单击“相加”按钮，其执行结果如图 3.15 所示。



图 3.14　webform2 网页执行界面一



图 3.15　webform2 网页执行界面二

从本例看出，单文件页模型只对应一个文件，即 webform2.aspx。其中，包含可视元素，如 Label1、TextBox1 和 Button1 等；又包含编程逻辑，如 Button1 的单击事件处理方法 Button1_Click，这些编程逻辑代码放置在<script>元素中。

2. 代码隐藏页模型

代码隐藏模型的特点是，在一个文件(*.aspx)中存放可视元素代码，在另一个文件(*.aspx.cs)中存放编程逻辑代码。代码文件的名称会根据所使用的编程语言而有所变化。

这种模型的优点是：可以清楚地分隔标记(HTML 代码)和代码(编程逻辑)，适用于包含大量代码或多个开发人员共同创建网站的 Web 应用程序；便于程序的维护和升级；程序代码可在多个网页中重用等。

【练一练】　在 CH3 网站中添加一个代码隐藏模型的网页 webform3，其功能与 webform2 网页相同。设计步骤如下：

① 进入 CH3 网站，选择“网站|添加新项”命令，出现的“添加新项”对话框，在中间列表中

选择“Web 窗体”，将文件名称改为 webform3. aspx，保留“将代码放在单独的文件中”的勾选，单击“添加”按钮，这样在 CH3 网站中添加一个名称为 webform3. aspx 的代码隐藏模型网页。

② 其他操作与 webform2 网页设计相同。

③ 单击 源 选项卡，看到 webform3 网页的源视图代码如下：

```
<%@ Page Language = "C#" AutoEventWireup = "true" CodeFile = "webform3.aspx.cs"
    Inherits = "webform3"  %>
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head runat = "server">
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8"/>
    <title></title>
    <style type = "text/css">
        .auto-style1 {
            font-family: 楷体;
            font-weight: bold;
            font-size: medium;
            color: #0000FF;
        }
    </style>
  </head>
  <body>
    <form id = "form1" runat = "server">
      <div>
        <asp:Label ID = "Label1" runat = "server" Text = "整数 1"
            CssClass = "auto-style1"></asp:Label>
        <asp:TextBox ID = "TextBox1" runat = "server" Width = "74px"></asp:TextBox>
          
        <asp:Label ID = "Label2" runat = "server" Text = "整数 2"
            CssClass = "auto-style1"></asp:Label>
        <asp:TextBox ID = "TextBox2" runat = "server" Width = "80px"></asp:TextBox>
      </div>
      <div>
        <br />
             
        <asp:Button ID = "Button1" runat = "server" Text = "相加" OnClick = "Button1_Click"
        style = "color:#FF0000;font-size:medium;font-weight:700;font-family:黑体" />
      </div>
      <div>
       <br />
       <asp:Label ID = "Label3" runat = "server"
        style = "color:#FF00FF;font-size:medium;font-weight:700;font-family:仿宋" />
      </div>
    </form>
  </body>
</html>
```

④ 在“解决方案资源管理器”中双击 webform3. aspx. cs 文件，看到其代码如下：

```
using System:
public partial class webform3 : System.Web.UI.Page
{     protected void Page_Load(object sender, EventArgs e)
```

```
        {
        }
        protected void Button1_Click(object sender, EventArgs e)
        {   int a = int.Parse(TextBox1.Text.Trim());
            int b = int.Parse(TextBox2.Text.Trim());
            int c = a + b;
            Label3.Text = "两个整数相加结果：" + c.ToString();
        }
}
```

webform3 网页的执行与 webform2 完全相同。

从中看到，代码隐藏模型的 webform3 网页的文件结构如图 3.16 所示，网页的可视元素和编程逻辑分离存放。在 webform3.aspx 文件的头部包含如下页面指令：

```
<%@ Page Language = "C#" AutoEventWireup = "true" CodeFile = "webform3.aspx.cs"
   Inherits = "webform3" %>
```

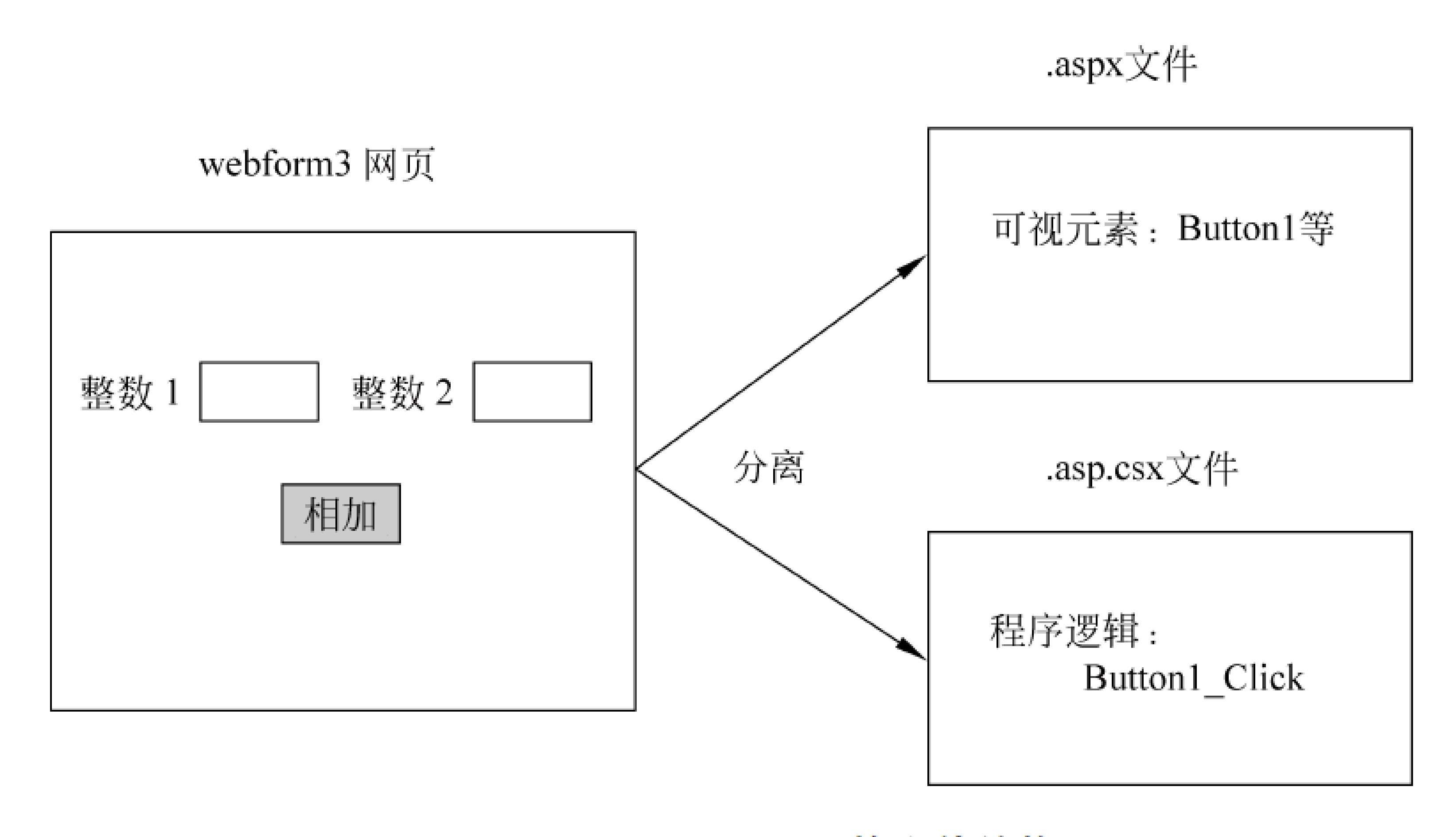


图 3.16　webform3 网页的文件结构

其作用是，指定 ASP .NET 网页编译器使用 C# 作为网页的服务器端代码语言，自动绑定网页的事件，该网页代码隐藏文件为 webform3.aspx.cs，供页面继承的代码隐藏类为 webForm3。也就是说，通过 CodeFile 和 Inherits 属性将分离存放的两个文件关联。

说明：对于代码隐藏模型的网页，即使不设计任何编程逻辑代码，如前面的 webform1 网页，也是采用两个文件存储的。

3.2.4　ASP .NET 网页的基本结构

ASP .NET 网页的基本结构是模块化的，每个 aspx 网页文件一般包含 3 个独立部分的部分：页面指令、代码脚本块和页面内容。

1. 页面指令

页面指令用于设置该页面的运行环境，规定 ASP .NET 如何处理该页面，控制 ASP .NET 页面的行为，一个页面可包含多条页面指令。页面指令不作为发送到浏览器标记的一部分呈现。当使用页面指令时，虽然标准的做法是将指令写在文件的开头，但是它们可以位于 .aspx 或 .ascx 文件中的任何位置。每个指令都可以包含一个或多个特定于该指令的属性（与值成对出现）。常见的 ASP .NET 页面指令如表 3.1 所示。

表 3.1　常见的页面指令

页 面 指 令	说　　明
@Page	定义 ASP .NET 页分析器和编译器使用的页(.aspx 文件)特定属性
@Import	将命名空间显式导入网页中,使所导入的命名空间的所有类和接口可用于该网页页。导入的命名空间可以是 .NET Framework 类库或用户自定义的命名空间的一部分
@OutputCache	以声明的方式控制 ASP .NET 页或页中包含的用户控件的输出缓存策略
@Implements	指示当前或用户实现指定的 .NET Framework 接口
@Register	将别名与命名空间及类名关联,以便在自定义服务器控件语法中使用简明的表示法
@Assembly	在编译过程中将程序集链接到当前页,以使程序集的所有类和接口都可用在该页上
@Control	定义 ASP .NET 页分析器和编译器使用的用户控件(.ascx 文件)特定属性。该指令只能用于用户控件
@Master	标识 ASP .NET 母版页
@MasterType	为 ASP .NET 网页的 Master 属性分配类名,使该页可以获取对母版页成员的强类型引用
@PreviousPageType	提供用于获得上一网页的强类型的方法,可通过 PreviousPage 属性访问上一网页
@Reference	以声明的方式指示,应该根据在其中声明此指令的网页对另一个用户控件或页源文件进行动态编译和链接

其中最常用的是@ Page 指令。该指令允许开发人员为网页指定多个配置选项,并且该指令只能在 Web 窗体网页中使用。每个.aspx 文件只能包含一条@Page 指令。该指令可以指定页面中代码的服务器编程语言、页面是将服务器代码直接包含在其中(即单文件页面),还是将代码包含在单独的类文件中(即代码隐藏页面)、调试和跟踪选项、页面是否为某母版页的内容页等。

其基本格式如下:

```
<%@ Page 属性 = "值" [属性 = "值" … ] %>
```

@Page 指令的属性及其说明如表 3.2 所示。若要定义@Page 指令的多个属性,需要使用一个空格分隔每个属性/值对。对于特定属性,不要在该属性与其值相连的等号(=)两侧加空格。

表 3.2　@Page 指令的属性及其说明

属　　性	说　　明
AutoEventWireup	指示页面的事件是否自动绑定。如果启用了事件自动绑定,则为 true(默认值),否则为 false
Buffer	确定是否启用了 HTTP 响应缓冲。如果启用了页面缓冲,则为 true(默认值),否则为 false
ClassName	一个字符串,指定在请求页面时将自动进行动态编译的页面的类名。此值可以是任何有效的类名,并且可以包括类的完整命名空间(完全限定的类名)。如果未指定该属性的值,则已编译页面的类名将基于页面的文件名
CodeBehind	指定包含页面关联类的已编译文件的名称,该属性不能在运行时使用
CodeFile	指定指向页面引用的代码隐藏文件的路径

续表

属　性	说　明
CodePage	指示用于响应的编码方案的值，该值是一个用作编码方案ID的整数
Description	提供该页面的文本说明，ASP .NET分析器忽略该值
ErrorPage	定义在出现未处理页面异常时用于重定向的目标URL
Inherits	定义供页面继承的代码隐藏类，与CodeFile属性（包含指向代码隐藏类的源文件的路径）一起使用
Language	指定在对页面中的所有内联呈现和代码声明块进行编译时使用的语言。只可以表示任何 .NETFramework支持的语言，如C＃
MasterPageFile	设置内容页面的母版页面或嵌套母版页面的路径，支持相对路径和绝对路径
Title	指定在响应的HTML＜title＞标记中呈现的页面的标题，也可以通过编程方式将标题作为页面的属性来访问
Trace	指示是否启用跟踪。如果启用了跟踪，则为true，否则为false（默认值）

例如，在前面的webform3. aspx网页文件中就看到了@ Page页面指令的作用。

2. 代码脚本块

代码脚本块是由“＜script runat＝server＞＜/script＞”标记对括起来的程序代码。在代码脚本块中可以定义页面的全局变量及控件事件处理程序等，这些程序代码要先编译后执行。代码脚本块采用C＃和Javascript等编写。

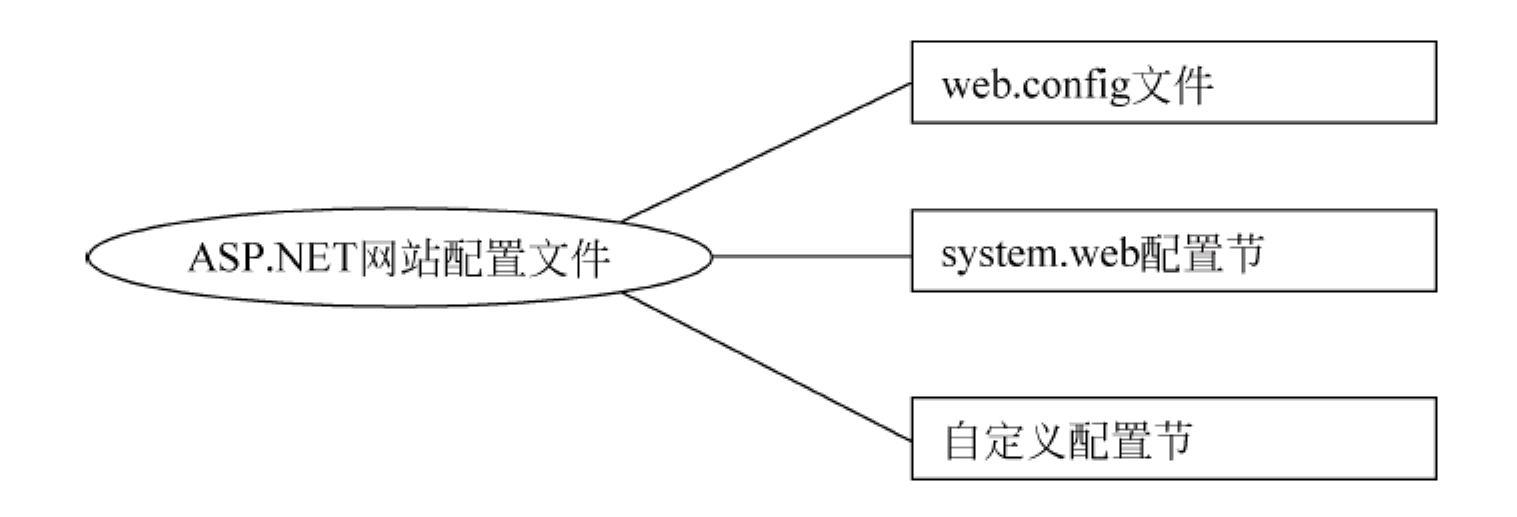
如果采用单文件页模型，代码脚本块放置在“＜script runat＝server＞…＜/script＞”中。如果采用代码隐藏页模型，代码脚本块放置在单独的文件中。

3. 页面内容

ASP .NET网页的页面内容是由ASP .NET控件构成的，有关ASP .NET控件的内容在第7章介绍。在一个网页中合理地布局ASP .NET控件是十分重要的。

3.3　ASP .NET网站配置文件

知识梳理



3.3.1　web. config文件

在创建一个网站时，总会在网站根目录中自动建立一个web. config文件，它就是网站的配置文件。

web. config文件是一个XML（可扩展标记语言）文本文件。XML与HTML相似，同属

于标记语言，但和 HTML 不同，XML 被设计用来传输和存储数据，其焦点是数据的内容；而 HTML 被设计用来显示数据，其焦点是数据的外观。另外，在 HTML 中，所有标记是预先定义的，每个标记都有特定的功能；而 XML 中标记并没有被预先定义，需要自行定义标记。

web.config 文件用来存储 ASP .NET 应用程序的配置信息（如最常用的设置 ASP .NET 应用程序的身份验证方式）。实际上，web.config 文件可以出现在应用程序的每一个目录中，根目录中的 web.config 文件包括默认的配置设置，所有的子目录都继承它的配置设置。

如果想修改子目录的配置设置，可以在该子目录下新建一个 web.config 文件，它可以提供除从父目录继承的配置信息以外的配置信息，也可以重写或修改父目录中定义的设置。

web.config 文件的根节点是<configuration>，在<configuration>节点下的常见子节点有<system.web>、<appSettings>和<connectionStrings>。这是一种嵌套结构，每个子节点中又可以包含若干子节点。

例如，最基本的 web.config 文件的内容如下：

```
<?xml version = "1.0"?>
<! --
  有关如何配置 ASP .NET 应用程序的详细信息，请访问
  http://go.microsoft.com/fwlink/?LinkId = 169433
  -->
< configuration >
  < system.web >
    < compilation debug = "true" targetFramework = "4.5"/>
    < httpRuntime targetFramework = "4.5"/>
  </system.web >
</configuration >
```

3.3.2　system.web 配置节

<system.web>节主要是网站运行时的一些配置，它的常见节点如下：

1. <compilation>节

<compilation>节用于配置 ASP .NET 使用的编译设置。默认的 debug 属性为 true，即允许调试，在这种情况下会影响网站的性能，所以在程序编译完成交付使用之后应将其设为 false。例如，若要禁止调试，设置<compilation>节如下：

```
< compilation
    debug = "false" >
</compilation >
```

2. <authentication>节

<authentication>节用于为 ASP .NET 应用程序配置 ASP .NET 身份验证方案。身份验证方案确定如何识别要查看 ASP .NET 应用程序的用户。其 mode 属性指定身份验证方案，它是必选的属性，其取值如表 3.3 所示。

其子元素有，Forms 为基于窗体的自定义身份验证配置 ASP .NET 应用程序；passport 指定要重定向到的页（如果该页要求身份验证，而用户尚未通过 Microsoft Passport Network 身份验证注册）。

表 3.3 mode 属性的取值及其说明

取　　值	说　　明
Windows(默认值)	将 Windows 验证指定为默认的身份验证模式。将它与以下任意形式的 Microsoft Internet 信息服务((IIS)身份验证结合起来使用：基本、摘要、集成 Windows 身份验证(NTLM/Kerberos)或证书。在这种情况下的应用程序将身份验证责任委托给基础 IIS
Forms	将 ASP .NET 基于窗体的身份验证指定为默认身份验证模式
Passport	将 Microsoft Passport Network 身份验证指定为默认身份验证模式
None	不指定任何身份验证。应用程序仅期待匿名用户，否则它将提供自己的身份验证

例如，以下示例为基于窗体(Forms)的身份验证配置站点，当没有登录的用户访问需要身份验证的网页，网页自动跳转到登录网页：

```
<authentication mode = "Forms">
    <forms loginUrl = "login.aspx" name = "FormsAuthCookie"/>
</authentication>
```

其中，元素 loginUrl 表示登录网页的名称；name 表示 Cookie 名称。

3. <authorization>节

<authorization>节用于控制对 URL 资源的客户端访问(如允许匿名用户访问)。此元素可以在任何级别(计算机、站点、应用程序、子目录或网页)上声明，与<authentication>节配合使用。其子元素有，allow 向授权规则映射添加一个规则，该规则允许对资源进行访问；deny 向授权规则映射添加一条拒绝对资源访问的授权规则。

例如，以下示例禁止匿名用户的访问：

```
<authorization>
    <deny users = "?"/>
</authorization>
```

其中，users 属性指出可访问的账号；? 表示以匿名方式访问；* 表示多有用户。

4. <customErrors>节

<customErrors>节用于为 ASP .NET 应用程序提供有关自定义错误信息的信息，不适用于 Web 服务中发生的错误。其子元素为 error(可选)，用于指定给定 HTTP 状态代码的自定义错误页。其属性及其说明如表 3.4 所示。

表 3.4 <customErrors>节的属性及其说明

属　　性	说　　明
defaultRedirect	可选的属性。指定出错时将浏览器定向到的默认 URL。如果未指定该属性，则显示一般性错误。URL 可以是绝对的(如 www.contoso.com/ErrorPage.htm)或相对的。相对 URL(如/ErrorPage.htm)是相对于为该属性指定 URL 的 web.config 文件，而不是相对于发生错误的网页。以波形符(～)开头的 URL(如～/ErrorPage.htm)表示指定的 URL 是相对于应用程序的根路径
mode	必选的属性。指定是启用或禁用自定义错误，还是仅向远程客户端显示自定义错误。此属性可以为下列值之一。On：指定启用自定义错误，如果未指定 defaultRedirect，用户将看到一般性错误，会向远程客户端和本地主机显示自定义错误。Off：指定禁用自定义错误，会向远程客户端和本地主机显示详细的 ASP .NET 错误。RemoteOnly(默认值)：指定仅向远程客户端显示自定义错误并且向本地主机显示 ASP .NET 错误

例如，在以下配置中，当网站运行发生错误时将自动跳转到自定义的错误网页 ErrorPage. aspx：

```
<customErrors defaultRedirect = "ErrorPage.aspx" mode = "RemoteOnly"> </customErrors>
```

3.3.3　自定义配置节

可以在<appSettings>节和<connectionStrings>节设置一些应用程序的设置项，即自定义配置。

1. <appSettings>节

此节用于定义应用程序设置项。对一些不确定设置，还可以让用户根据实际情况设置。例如，在其中添加用于存储数据库连接字符串的子节点。当然，如果程序需要其他自定义的全局配置信息，也可以在此添加相应的子节点。

appSettings 元素的子元素有：add(可选的子元素)向应用程序设置集合添加名称/值对形式的自定义应用程序设置；clear(可选的子元素)移除所有对继承自定义应用程序设置的引用，仅允许由当前 add 属性添加的引用；remove(可选的子元素)从应用程序设置集合中移除对继承自定义应用程序设置的引用。

例如，在 web. config 文件中的<appSettings>节中，采用<add>添加了一个与 SQL Server 数据库 Stud 连接的子节点和一个 Web 服务子节点：

```
<appSettings>
    <remove key = "mydata" />
    <add key = "mydata" value = "我的数据" />
</appSettings>
```

对于<appSettings>节中的子节点，可以使用 ConfigurationManager. AppSettings["key值"]来读取这些子节点值。例如：

```
mystr = System.Configuration.ConfigurationManager.
   AppSettings["myconnstring"].ToString();
```

其中，ConfigurationManager 类位于 System. Configuration 命名空间。这样，等价于如下语句：

```
mystr ="我的数据";
```

2. <connectionStrings>节

在 ASP .NET 中，如会话、成员资格、个性化设置和角色管理器等功能均依赖于存储在 connectionStrings 元素中的连接字符串，还可以使用 connectionStrings 元素来存储应用程序的连接字符串。

connectionStrings 元素的子元素有：add 子元素向连接字符串集合添加名称/值对形式的连接字符串；clear 子元素移除所有对继承的连接字符串的引用，仅允许那些由当前 add 元素添加的连接字符串；remove 子元素从连接字符串集合中移除对继承连接字符串的引用。

例如，在 web. config 文件中的<connectionStrings>节中，采用<add>添加了一个与 SQL Server 数据库 school 连接的子节点：

```
connectionStrings>
    <add name = "myconnstring"
       connectionString = "Data Source = LCB - PC;Initial Catalog = school;
          Integrated Security = True"
    providerName = "System.Data.SqlClient" />
  </connectionStrings>
```

对于<connectionStrings>节中的子节点的 Web 应用程序配置信息，可以使用 ConfigurationManager.ConnectionSettings["key 值"].ToString()来读取这些子节点值。例如：

```
mystr = System.Configuration.ConfigurationManager.
          ConnectionStrings["myconnstring"].ToString();
```

这样，等价于如下语句：

```
mystr = "Data Source = LCB - PC;Initial Catalog = school;Integrated Security = True";
```

如果整个网站中只有一个网页需要使用 myconnstring，这样做的意义不大，但若有多个网页需要，这样设计就十分必要。当 myconnstring 发生更改时，只需要更改<connectionStrings>节中相应的定义即可。

3.4　练　习　题

1．单项选择题

(1) 创建 ASP .NET 网站时，“新建网站”对话框中“Web 位置”选项不能是(　　)。

A．文件系统　　B．HTTP　　C．FTP　　D．ASP .NET

(2) 文件系统网站适合于学习使用，因为(　　)。

A．不需要安装 IIS　　B．网站允许放置在任意目录下

C．能够进行单独测试　　D．A 和 B

(3) 以下叙述中正确的是(　　)。

A．一个 ASP .NET 网站至少包含一个网页文件

B．一个 ASP .NET 网站至少包含一个数据库文件

C．一个 ASP .NET 网站至少包含一个应用程序类文件

D．一个 ASP .NET 网站至少包含一个图像文件

(4) 采用“ASP .NET 空网站”模板创建一个空网站时，该网站包含(　　)。

A．一个 Default.aspx 网页文件

B．一个配置文件(web.config)

C．一个 SheetStyle.css 样式文件

D．一个 SkinFile.skin 外观文件

(5) 一个 ASP .NET 网站包含若干目录，通常数据库文件放在(　　)目录中。

A．App_Data　　B．App_Code　　C．App_Themes　　D．Bin

(6) 一个 ASP .NET 网站包含若干目录，通常应用程序的类文件放在(　　)目录中。

A．App_Data　　B．App_Code　　C．App_Themes　　D．Bin

(7) ASP .NET 网页开发有单文件页模型和代码隐藏页模型两种，以下叙述正确的是(　　)。

A. 代码隐藏页模型的网页开发功能更强，而单文件页模型的网页开发功能较弱

B. 单文件页模型的网页开发功能更强，而代码隐藏页模型的网页开发功能较弱

C. 两种在网页开发上功能是等价的

D. 以上都不对

(8) ASP .NET 网页开发有单文件页模型和代码隐藏页模型两种，以下叙述错误的是(　　)。

A. 单文件页模型中，C＃代码必须包含于＜script＞…＜/script＞之间

B. 代码隐藏页模型中，可视元素和编程逻辑代码放在一个文件中

C. 代码隐藏页模型中，可视元素和编程逻辑代码放在不同文件中

D. 单文件页模型中，可视元素和编程逻辑代码放在不同文件中

(9) 采用 Visual Studio 设计的网页代码称为 ASP .NET 网页代码，最终浏览器运行的网页代码称为纯 HTML 代码。以下叙述正确的是(　　)。

A. 在用户请求 ASP .NET 网页时，需要 ASP .NET 引擎将 ASP .NET 网页代码转换为纯 HTML 代码

B. 不需要 ASP .NET 引擎，Web 服务器就会将 ASP .NET 网页代码转换为纯 HTML 代码

C. ASP .NET 网页代码和纯 HTML 代码是一样的

D. 以上都不对

(10) 用户通过 Internet 请求 ASP .NET 网页，对应的 Web 服务器(　　)。

A. 只需要配置 IIS 即可

B. 只需要配置 ASP .NET 引擎即可

C. 必须配置 IIS 和 ASP .NET 引擎

D. 以上都不对

(11) ASP .NET 的@ Page 指令的作用是(　　)。

A. 定义 ASP .NET 页分析器和编译器使用的页的特定属性

B. 将命名空间显式导入到网页中

C. 以声明的方式控制 ASP .NET 页或页中包含的用户控件的输出缓存策略

D. 指示当前或用户实现指定的 .NET Framework 接口

(12) ASP .NET 的@ Import 指令的作用是(　　)。

A. 定义 ASP .NET 页分析器和编译器使用的页的特定属性

B. 将命名空间显式导入到网页中

C. 以声明的方式控制 ASP .NET 页或页中包含用户控件的输出缓存策略

D. 指示当前或用户实现指定的 .NET Framework 接口

(13) 下列选项中，只有(　　)不是@ Page 指令的属性。

A. CodeBehind　　B. Buffer　　C. NameSpace　　D. Language

(14) ASP .NET 的@ Page 指令中 CodeFile 属性的含义是(　　)。

A. 指定包含与页面关联的类的已编译文件的名称

B. 指定指向页面引用的代码隐藏文件的路径

C. 指示用于响应的编码方案的值，该值是一个用做编码方案 ID 的整数

D. 定义供页面继承的代码隐藏类

(15) ASP.NET的@ Page指令中Inherits属性的含义是(　　)。

A. 指定包含与页面关联的类的已编译文件的名称

B. 指定指向页面引用的代码隐藏文件的路径

C. 指示用于响应的编码方案的值,该值是一个用做编码方案ID的整数

D. 定义供页面继承的代码隐藏类

(16) web.config文件不能用于(　　)。

A. Application事件定义　　B. 数据库连接字符串定义

C. 对文件夹访问授权　　D. 基于角色的安全性控制

2. 问答题

(1) 简述创建一个ASP.NET空网站的步骤。

(2) 简述开发ASP.NET网页的单文件页模型和代码隐藏页模型的区别。

(3) 网页设计窗口下方有 设计 、 源 和 拆分 3个选项卡,说明它们的作用。

(4) 在一个ASP.NET网页中有如下源视图代码:

```
<asp:TextBox ID = "TextBox1" runat = "server"></asp:TextBox>
```

给出对应的纯HTML代码。

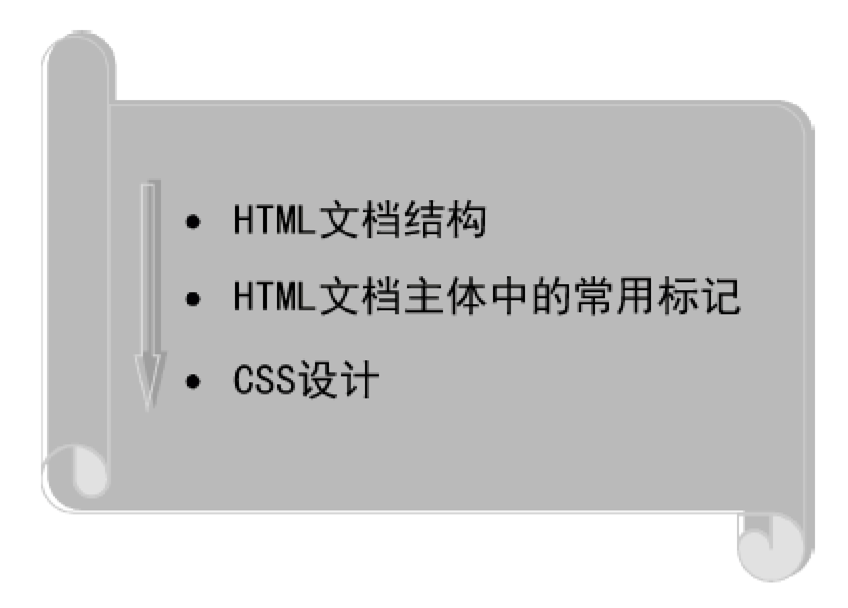
(5) 简述页面指令@ Page的作用。

(6) 简述web.config文件中<connectionStrings>节的作用。

第 4 章　HTML 和 CSS

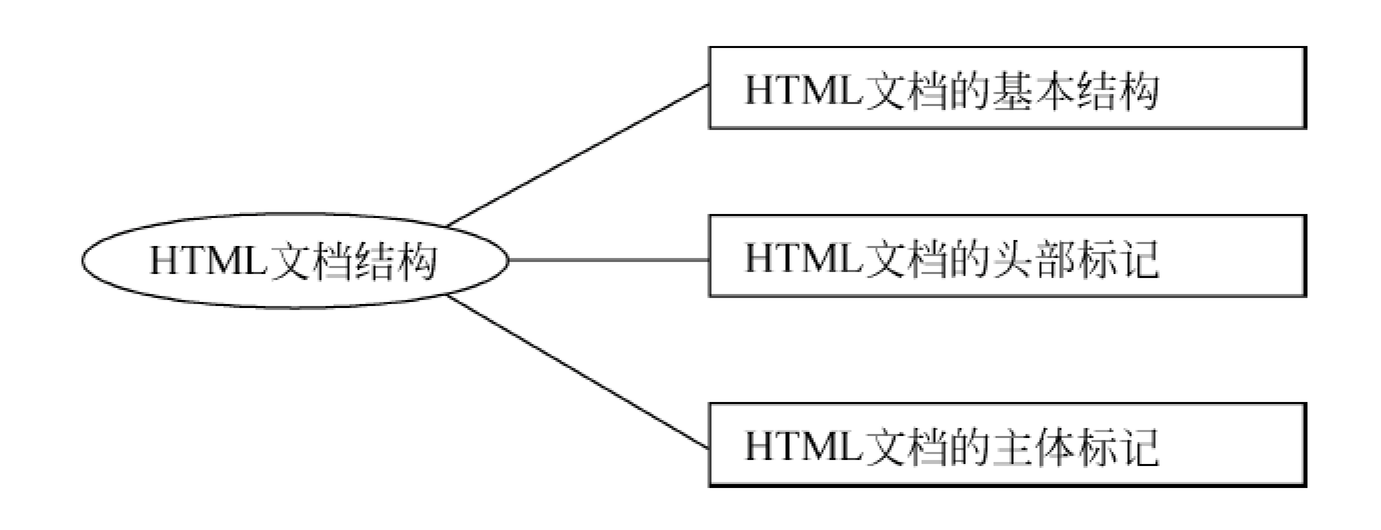


本章指南



4.1　HTML 文档结构

知识梳理



4.1.1　HTML 文档的基本结构

HTML 文档就是网页，是一种普通文本文件。网页可以是网站的一部分，也可以独立存在，可以用记事本等文本编辑器进行编辑，本书使用 Visual Studio 来编辑 HTML 文档。纯 HTML 文件的扩展名为 html 等。

从结构上看，HTML 文档一般分为 DOCTYPE、文档头部（head）和文档主体（body）3 个

部分，其基本结构如图4.1所示。

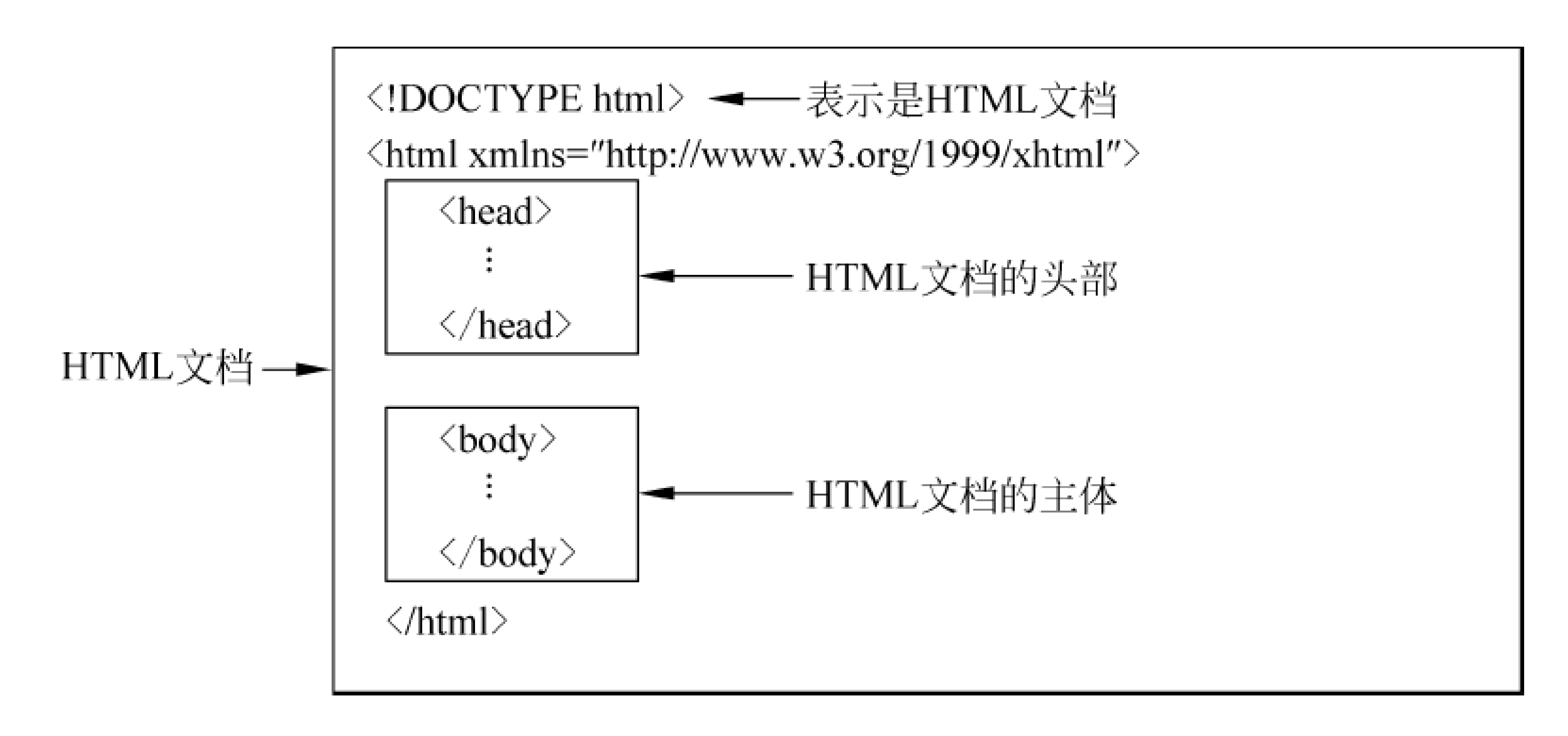


图4.1　HTML文档的基本结构

其中，DOCTYPE是文档类型的声明，向系统声明当前网页的文档类型，浏览器根据这个声明对HTML代码做出相应的处理，不同的声明会有不同的处理。这些处理可能会影响HTML标记的表现，也就是说，同一段HTML代码，不同的声明可能会得到不同的页面效果。

DOCTYPE标记不需要闭合，必须放在＜html＞元素之前。除了DOCTYPE声明外，其他所有HTML代码都包含在＜html＞和＜/html＞标记之间。

XHTML(可扩展超文本标记语言)也是一种置标语言，表现方式与HTML类似，不过语法上更加严格。XHTML可以说就是HTML一个升级版本。Visual Studio支持XHTML。

在XHTML中，＜html＞的xmlns属性是必需的，它定义XML的命名空间。不过，即使XHTML文档中的＜html＞没有使用此属性，W3C的验证器也不会报错。这是因为“xmlns=http://www.w3.org/1999/xhtml”是一个固定值(W3C的默认命名空间)，即使没有包含它，此值也会添加到＜html＞标记中。

4.1.2　HTML文档的头部标记

HTML头部一般用于标记文档的某些信息，放于＜head＞与＜/head＞之间。它们通常不会显示在网页上。用于HTML头部的标记有以下几种：＜title＞和＜meta＞等。其中，＜title＞是任何网页必须有的，其他均为可选项。

1. title标记

基本用法：

```
<title> … </title>
```

该标记用于指定文档的标题，通常＜title＞…＜/title＞中间的文字会显示在浏览器的标题栏上。一般用简明扼要的文字概括该文档的主要内容或主题，不超过64个ASCII码字符长度，如果过长，窗口的标题栏无法容纳。

2. meta标记

基本用法：

```
<meta 属性="属性值"/>
```

该标记用于描述网页的具体摘要信息，包括文档的内容类型、字符编码信息、搜索关键字、网站提供的功能和服务的详细描述等。其内容并不在浏览器中显示。其中，http-equiv 属性把 content 属性关联到 HTTP 头部；name 属性把 content 属性关联到一个名称；content 属性定义与 http-equiv 或 name 属性相关的元信息（必选）；charset 属性定义文档的字符编码。例如，以下语句指定若干关键字：

```
<meta name="keywords" content="Visual Studio,ASP .NET,动态网页设计">
```

又如，以下语句向浏览器报告本文档作者是李兵：

```
<meta name="作者" content="李兵">
```

还如，以下语句表示文档的内容类型为 html 类型，字符编码为国际通用的字符编码：

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
```

4.1.3　HTML 文档的主体标记

主体标记的基本用法如下：

```
<body> … </body>
```

<body>标记定义文档的主体。body 标记包含文档的所有内容（如文本、超链接、图像、表格和列表等）。

为了在浏览器中显示完整内容，需在网页中添加文本、图形、表格、表单、超链接等网页元素。HTML 语言通过各种标记控制这些网页元素的显示方式。一个含有 header、section 和 footer 元素网页文档主体部分的基本结构如下：

```
<body>
  <header>
   …<!--定义页面的页眉-->
  </header>
  <section>
    <!--定义节-->
  </section>
  <footer>
      <!--定义页面的页脚-->
  <footer>
</body>
```

【练一练】　在 D 盘的电子商务目录中建立一个 CH4 的子目录，将其作为网站目录，设计一个 webform1.html 网页，其功能是说明 HTML 头部和主体标记的应用。

其设计步骤如下：

① 启动 Visual Studio 2012。

② 选择“文件|新建|网站”命令，出现“新建网站”对话框，选择“ASP .NET 空网站”模板，选择“Web 位置”为“文件系统”，单击“浏览”按钮，选择“D:\电子商务\CH4”目录，单击“确定”按钮，创建了一个空的网站 CH4。

③ 选择“网站|添加新项”命令，出现“添加新项-CH4”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform1. html，如图 4. 2 所示，单击“添加”按钮。



图 4. 2　添加 webform1. html 网页

说明： 这种“HTML 页”模板类型的网页只能采用单文件页模型设计。

④ 进入源视图，设计其 HTML 代码如下(大部分是 Visual Studio 自动产生的)：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>HTML 文档</title>
  </head>
  <body>
    电子商务开发技术
  </body>
</html>
```

该网页在 IE 浏览器中的执行结果如图 4. 3 所示。

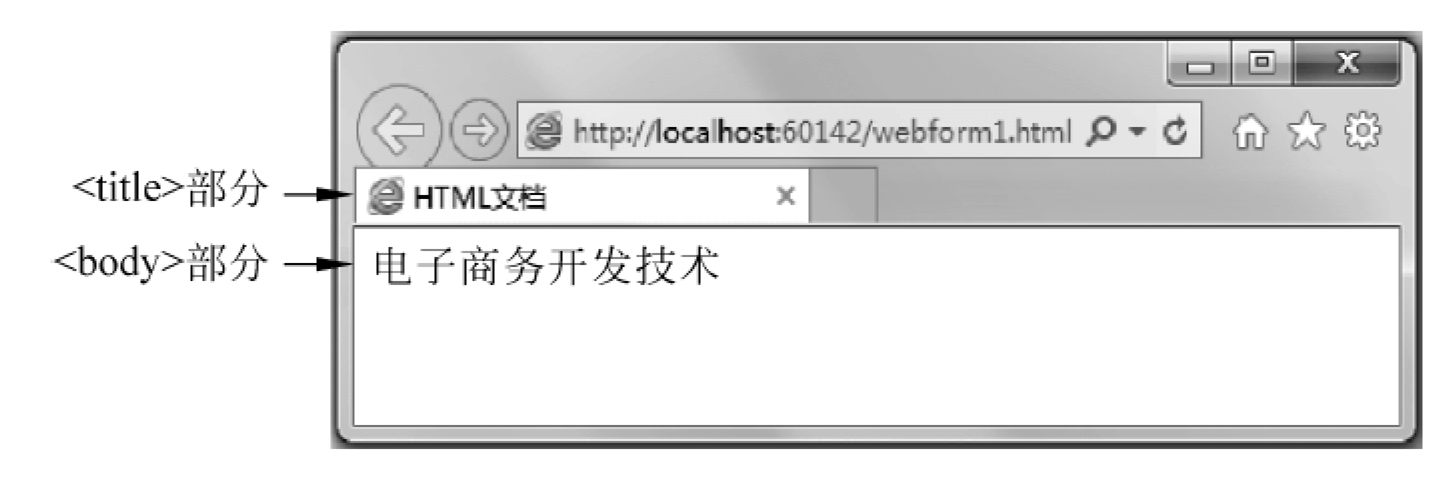
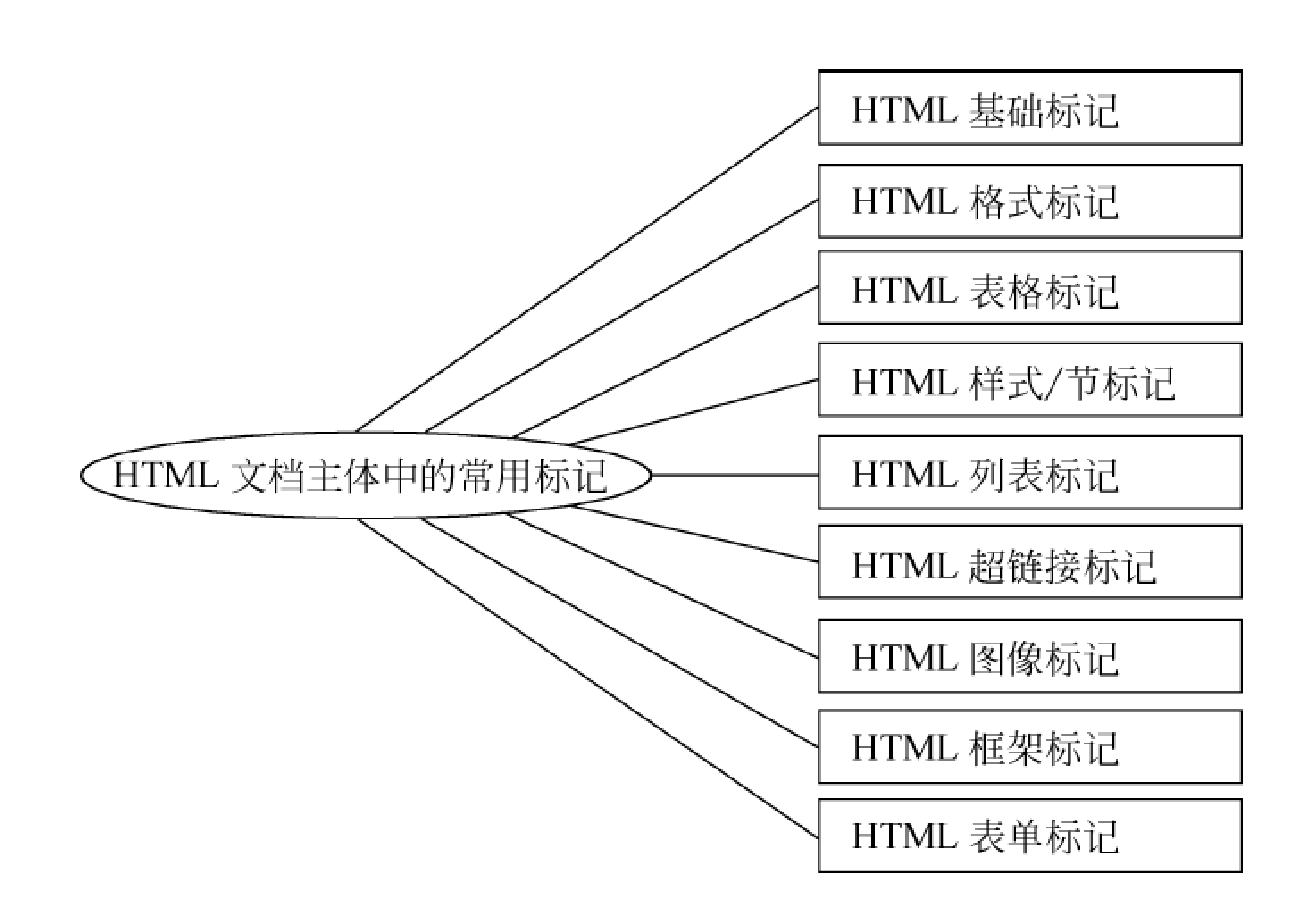


图 4. 3　webform1. html 网页执行界面

4.2　HTML 文档主体中的常用标记

知识梳理



4.2.1　HTML 基础标记

1. 标题标记

基本用法：

```
<hn> … </hn>  n = 1, 2, 3, 4, 5, 6
```

该标记确定字体的显示方式，按标题级别突出显示这些标题文字。字体从 h1～h6 逐级减小。可以使用 style 属性规定元素的行内样式，样式定义是一个或多个由分号分隔的 CSS 属性和值。

例如，以下代码以 h1 标题文字、蓝色和居中格式显示“中华人民共和国”：

```
<h1 style = "color:blue; text - align:center">中华人民共和国</h1>
```

2. 段落标记

基本用法：

```
<p> … </p>
```

该标记用于定义一个段落。可以忽略文档中原有的回车和换行来定义一个新段落，换行并插入一个空行。

3. 换行标记

基本用法：

```
<br />
```

该标记强行中断当前行，使后续文本在下一行显示。

4. 水平线标记

基本用法：

```
<hr />
```

该标记在文档中添加一条水平线，用来分隔文档。

5. 定义注释标记

基本用法：

```
<!-- … -->
```

注释标签用于在源代码中插入注释。注释不会显示在浏览器中。

【练一练】 在CH4网站中添加一个webform2.html网页，其功能是说明HTML基础标记的应用。

其设计步骤如下：

① 打开CH4网站，选择“网站|添加新项”命令，出现“添加新项-CH4”对话框，在中间列表中选择“HTML页”，将文件名称改为webform2.html，单击“添加”按钮。

② 进入源视图，设计其HTML代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
  </head>
  <body>
     <h2>早发白帝城</h2>
     <h4>李白</h4>
     <hr />
     <p>
       <h3>朝辞白帝彩云间<br />
         千里江陵一日还<br />
         两岸猿声啼不住<br />
         轻舟已过万重山
       </h3>
     </p>
  </body>
</html>
```



图4.4 webform2.html网页执行界面

该网页在IE浏览器中的执行结果如图4.4所示。

4.2.2 HTML格式标记

HTML中几个常见的格式标记如表4.1所示。

表 4.1　几个常见的格式标记及其说明

格式标记	功　　能
<tt>	呈现类似打字机或等宽的文本效果
<i>	显示斜体文本效果
<b>	呈现粗体文本效果
<em>	定义强调文本
<small>	呈现小号字体效果
<strong>	定义语气更为强烈的强调文本
<sub>	定义下标文本
<sup>	定义上标文本
<mark>	定义有记号的文本
<meter>	定义预定义范围内的度量
<time>	定义日期/时间

【练一练】 在 CH4 网站中添加一个 webform3. html 网页，其功能是说明常见的格式标记的应用。

其设计步骤如下：

① 打开 CH4 网站，选择“网站|添加新项”命令，出现“添加新项-CH4”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform3. html，单击“添加”按钮。

② 进入源视图，设计其 HTML 代码如下：

```
<! DOCTYPE html >
< html xmlns = "http://www.w3.org/1999/xhtml">
  < head >
    < meta http - equiv = "Content - Type" content = "text/html; charset = utf - 8"/>
    < title ></title >
  </head >
  < body >
     < b >粗体文本</b >
     < br />
     < em >< mark >强调文本</mark ></em >
     < br />
     < strong >< mark >强烈的强调文本</mark ></strong >
     < br />
     < i >斜体文本</i >
     < br />
     < small >小号字体文本</small >
     < br />
     A < sub >下标文本</sub >
     < br />
     A < sup >上标文本</sup >
     < br />
     我们在每天早上< time > 9:00 </time >点开始营业
  </body >
</html >
```



图 4.5　webform3.html 网页执行界面

该网页在 IE 浏览器中的执行结果如图 4.5 所示。

4.2.3　HTML 表格标记

表格是人们处理数据最常用的一种形式，一个表格由表标题、行、列标题和单元格组成。表格不仅可以用来罗列数据，还可以将文本、图像、超链接等各种对象放到表格中进行定位，从而制作出排版精美的网页。常见的表格标记如表 4.2 所示。

表 4.2　几个常见的表格标记及其说明

表格标记	功　　能
＜table＞	定义表格
＜caption＞	定义表格标题
＜th＞	定义表格中的表头单元格
＜tr＞	定义表格中的行
＜td＞	定义表格中的单元
＜thead＞	定义表格中的表头内容
＜tbody＞	定义表格中的主体内容
＜tfoot＞	定义表格中的表注内容(脚注)
＜col＞	定义表格中一个或多个列的属性值
＜colgroup＞	定义表格中供格式化的列组

在设计表格时常用的属性如下：

对于＜table＞标记，border 表示是否有边框，为"0"表示没有边框，为"1"表示有边框；cellpadding 规定单元边沿与其内容之间的空白；cellspacing 规定单元格之间的空白。

对于单元格，colspan＝"2"表示横跨两列的单元格，rowspan＝"2"表示横跨两行的单元格；background 设置单元格背景图像；bgcolor 设置单元格背景颜色。

另外，在一个表格中可以嵌套另一个表格。

【练一练】　在 CH4 网站中添加一个 webform4. html 网页，其功能是说明表格标记的应用。

其设计步骤如下：

① 打开 CH4 网站，选择“网站|添加新项”命令，出现“添加新项-CH4”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform4. html，单击“添加”按钮。

② 进入源视图，设计其 HTML 代码如下：

```
<! DOCTYPE html >
< html xmlns = "http://www.w3.org/1999/xhtml">
  < head >
     < meta http - equiv = "Content - Type" content = "text/html; charset = utf - 8"/>
     < title ></title >
  </head >
  < body >
    < table border = "1" style = "text - align:center;width:300px;border:double">
      < caption >< b >学生情况表</b ></caption >
      < tr >
        < th >学号</th >
        < th >姓名</th >
        < th >性别</th >
        < th >班号</th >
      </tr >
      < tr >
        < td > 101 </td >
```

```
            <td>王华</td>
            <td>女</td>
            <td rowspan = "3"> 1501 </td>
        </tr>
        <tr>
            <td> 103 </td>
            <td>李民</td>
            <td>男</td>
        </tr>
        <tr>
            <td> 108 </td>
            <td>张丽</td>
            <td>女</td>
        </tr>
        <tr>
            <td> 112 </td>
            <td>陈强</td>
            <td>男</td>
            <td rowspan  = "2"> 1502 </td>
        </tr>
        <tr>
            <td> 138 </td> <td>李兵</td>
            <td>男</td>
        </tr>
    </table>
  </body>
</html>
```

该网页在 IE 浏览器中的执行结果如图 4.6 所示。采用<table>标记创建一个表格,其标题为“学生情况表”,插入 6 行,每行 4 列,第 2、3、4 行的第 4 列是合并的(通过 rowspan="3"来实现),第 5、6 行的第 4 列也是合并的。



图 4.6　webform4.html 网页的执行界面

4.2.4　HTML 样式/节标记

HTML 中主要的样式/节标记如表 4.3 所示。

表 4.3　主要的样式/节标记及其说明

样式/节标记	功　能
<style>	定义文档的样式信息
<div>	定义文档中的块
<span>	定义文档中的节
<header>	定义 section 或 page 的页眉
<footer>	定义 section 或 page 的页脚
<section>	定义 section
<article>	定义文章
<aside>	定义页面内容之外的内容
<details>	定义元素的细节
<dialog>	定义对话框或窗口
<summary>	为<details>元素定义可见的标题

1. style

<style>标记用于为HTML文档定义样式信息。在style中可以规定在浏览器中如何呈现HTML文档。就style单词而言，既可以用作HTML标记，也可以用作HTML属性，要注意两者的区别。

2. <div>标记

基本用法：

```
<div 属性="属性值"> …  </div>
```

该标记用来定义文档中的块，可以把文档分割为独立的、不同的部分。它是内容自动地开始一个新行。

3. <section>标记

基本用法：

```
<section 属性="属性值"> … </section>
```

该标记可定义文档中的节(section)，如章节、页眉、页脚或文档中的其他部分。

说明：<div>、<section>和<article>等节标记通常用于将若干HTML元素组合在一起，可以采用绝对和相对定位方式进行页面布局，详细内容常见4.3.2小节。

【练一练】 在CH4网站中添加一个webform5.html网页，其功能是说明样式/节标记的应用。

其设计步骤如下：

① 打开CH4网站，选择“网站|添加新项”命令，出现“添加新项-CH4”对话框，在中间列表中选择“HTML页”，将文件名称改为webform5.html，单击“添加”按钮。

② 进入源视图，设计其HTML代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
  </head>
  <body>
    <div style="color:blue">
        <h2>要闻</h2>
        <p>"一带一路"十年目标：年贸易额2.5万亿
美元</p>
    </div>
    <section>
        <h2>体育新闻</h2>
        <p>发展足球绝不是说不开展别的体育活动
</p>
    </section>
  </body>
</html>
```



图4.7 webform5.html网页执行界面

该网页在IE浏览器中的执行结果如图4.7所示。采用<div>和<section>标记将所有显示的文

字分为两组。

4.2.5　HTML 列表标记

列表用于显示一组意义相似的列表项，每个列表项由一个文字串构成。列表分为有序、无序列表和自定义列表。

1. 有序列表

基本用法：

```
<ol start="起始序号" type="符号类型">
   <li>…</li>
   <li>…</li>
    ⋮
</ol>
```

其功能是建立有序列表。start 属性指出数字序列的起始值；type 属性指出数字序号的样式即符号类型，取值如下：

- 1：表示阿拉伯数字 1、2、3 等，此为默认值。
- a：表示小写字母 a、b、c 等。
- A：表示大写字母 A、B、C 等。
- i：表示小写罗马数字 i、ii、iii、iv 等。
- I：表示大写罗马数字 I、II、III、IV 等。

标记用于定义列表项，位于
和
之间。它有两个常用的属性：

- type：指出该列表项的符号类型(其取值与 ol 标记的 type 属性取值相同)。
- value：新的数定序列起始值以获取非连续的数字序列。

2. 无序列表

基本用法：

```
<ul type="项目符号">
   <li>…</li>
   <li>…</li>
    ⋮
</ul>
```

其功能是建立无序列表。每个列表项以无编号的形式列出，其前有一个项目符号，由属性 type 决定，type 的取值如下所示。

- disc：使用实心圆作为项目符号，此为默认值。
- circle：使用空心圆作为项目符号。
- square：使用方块作为项目符号。

3. 自定义列表

基本用法：

```
<dl>
   <dt>…</dt>
   <dd>…</dd>
   <dd>…</dd>
     …
```

```
</dl>
```

自定义列表属于描述性列表，用于表示信息的层次关系。

注意：根据文档的具体要求，自定义列表可以嵌套使用。

【练一练】 在 CH4 网站中添加一个 webform6. html 网页，其功能是说明列表标记的应用。

其设计步骤如下：

① 打开 CH4 网站，选择“网站|添加新项”命令，出现“添加新项-CH4”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform6. html，单击“添加”按钮。

② 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
     <meta http - equiv = "Content - Type" content = "text/html; charset = utf - 8"/>
     <title></title>
  </head>
  <body>
    <! -- 插入一个 1 行 3 列的表格 -->
    <table border = "1" style = "text - align:center;width:400px;border:double">
      <tr>
         <td>
           <h3>球类</h3>
           <ol start  = "1" type  =  "A">
             <li>足球</li>
             <li>篮球</li>
             <li>排球</li>
             <li>乒乓球</li>
           </ol>
        </td>
        <td>
          <h3>田径</h3>
          <ul>
             <li>跑步</li>
             <li>铅球</li>
             <li>跳高</li>
             <li>跳远</li>
          </ul>
        </td>
        <td>
          <h3>体育</h3>
          <dl>
            <dt>球类</dt>
               <dd>足球</dd>
               <dd>篮球</dd>
               <dd>排球</dd>
               <dd>乒乓球 </dd>
            <dt>田径</dt>
               <dd>跑步</dd>
               <dd>铅球</dd>
               <dd>跳高</dd>
               <dd>跳远</dd>
          </dl>
```

```
        </td>
      </tr>
    </table>
  </body>
</html>
```

该网页在 IE 浏览器中的执行结果如图 4.8 所示。采用<table>标记建立一个 1 行 3 列的表格，第 1 列插入一个有序列表，第 2 列插入一个无序列表，第 3 列插入一个自定义列表。

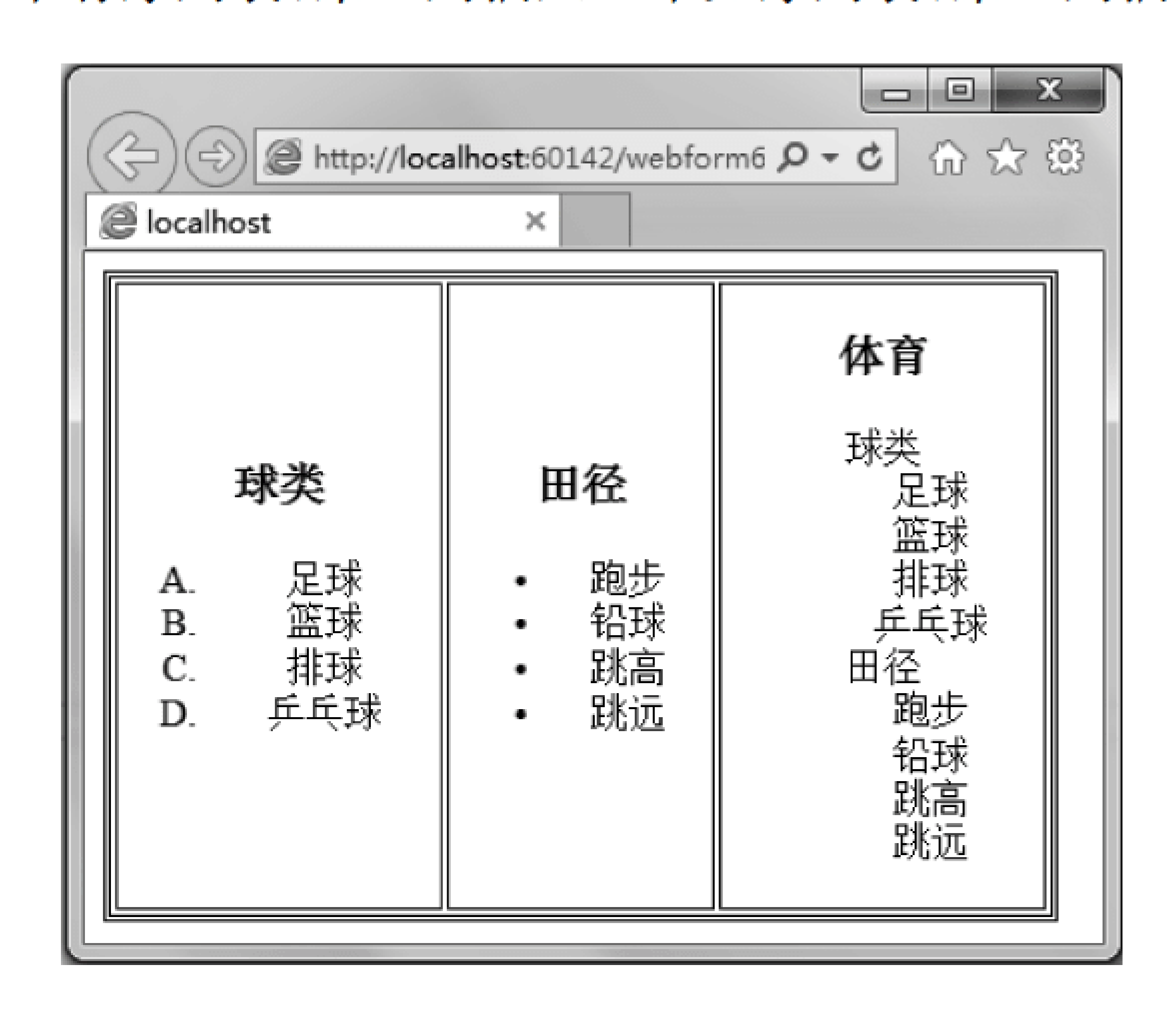


图 4.8 webform6.html 网页的执行界面

4.2.6 HTML 超链接标记

超链接用于实现浏览转向功能，是 Web 网页的基本功能之一，通过它将众多网页组织到一起。主要的超链接标记如表 4.4 所示。

表 4.4 主要的超链接标记及其说明

链接标记	功　　能
<a>	定义锚点
<link>	定义文档与外部资源的关系，最常见的用途是链接样式表，如<link rel="stylesheet" type="text/css" href="theme.css" />
<nav>	定义导航超链接。如果文档中有"前后"按钮，则应该把它放到<nav>标记中

超链接标记最常用的属性是 href，它指出转向的 URL。另外，target 属性指出该超链接指向的 HTML 文档在指定目标窗口中打开，target 属性取值及其说明如表 4.5 所示。

表 4.5 目标窗口名称及其说明

名　　称	说　　明
_blank	将超链接的内容显示在新的浏览器窗口中
_parent	将超链接的内容显示在父窗口中
_search	将超链接的内容显示在搜索窗口中
_self	将超链接的内容显示在当前窗口中
_top	将超链接的内容显示在浏览器主窗口中

<a>标记有以下两种基本用法。

1. 设置锚点(书签)

设置锚点的基本格式为：

```
<a name="锚点名称"> … </a>
```

锚点相当于书签，即在网页中的某一处设置一个标记，其他的超链接可以指向它。锚点通常用于链接到同一个网页的不同位置。

2. 设置超链接

超链接是网页中最重要的元素之一，是一个网站的灵魂。一个网站是由多个网页组成的，网页之间依靠超链接确定相互的导航关系，超链接使得网页的浏览非常方便。超链接的基本格式为：

```
<a href="url"> … </a>
```

其中，URL 可以使用绝对路径或相对路径。

【练一练】 在 CH4 网站中添加一个 webform7.html 网页，其功能是说明超链接标记的应用。

其设计步骤如下：

① 打开 CH4 网站，选择“网站|添加新项”命令，出现“添加新项-CH4”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform7.html，单击“添加”按钮。

② 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
  </head>
  <body>
    <p><h2><a href="http://www.whu.edu.cn/">武汉大学</a></h2></p>
    <p><a href="#C1">查看第 1 章</a></p>
    <p><a href="#C2">查看第 2 章</a></p>
    <p><a href="#C3">查看第 3 章</a></p>
    <p><a href="#C4">查看第 4 章</a></p>
    <h3><a name="C1">第 1 章</a></h3>
    <p>第 1 章内容...</p>
    <h3><a name="C2">第 2 章</a></h3>
    <p>第 2 章内容...</p>
    <h3><a name="C3">第 3 章</a></h3>
    <p>第 3 章内容...</p>
    <h3><a name="C4">第 4 章</a></h3>
    <p>第 4 章内容...</p>
  </body>
</html>
```

该网页在 IE 浏览器中的初始执行结果如图 4.9 所示。用户单击“武汉大学”超链接便转向武汉大学网站，单击“查看第 1 章”超链接便转向本网页的第 1 章锚点处，如图 4.10 所示。





图 4.9　webform7.html 网页执行界面一



图 4.10　webform7.html 网页执行界面二

4.2.7　HTML 图像标记

图像是最早引进 Web 页的多媒体对象，由于有了图像，Web 可以图文并茂地向用户提供信息，成倍地加大了它所提供的信息量，图像的引入也极大地美化了 Web 网页。Web 网页制作的很多技巧就是如何利用好图像，使网页美观匀称，可以使用图像标记链入图像。主要的 HTML 图像标记如表 4.6 所示。

表 4.6　主要的 HTML 图像标记及其说明

图像标记	功　能
<img>	定义图像
<map>	定义图像映射
<area>	定义图像地图内部的区域
<canvas>	定义图形
<figcaption>	定义 figure 元素的标题
<figure>	定义媒介内容的分组，以及它们的标题

1. <img>标记

<img>标记的基本用法：

```
< img 属性 = "属性值">
```

图像标记的必需属性如下：

- src：链接图像的 URL 位置，通常采用相对路径。例如，“../Images/logo.gif”就是一个相对路径，表示当前目录的上一级目录中 Images 目录中的 logo.gif 文件。图像可以是 jpeg、gif 或 png 文件。
- alt：该图形的信息，如文件名、文件大小和描述等。

注意：<img>标记并不会在网页中插入图像，而是从网页上链接图像。<img>标记创建的是被引用图像的占位空间。

2. 绝对 URL 和相对 URL

绝对 URL 是包含网站域名和协议信息(http://前缀)的完整路径。例如,以下就是武汉大学网站中一幅新闻图片的绝对 URL:

```
http://news.whu.edu.cn/_mediafile/whu_news/2015/04/14/2ppgcrb9dk.jpg
```

可以通过如下代码来引用它:

```
<img src="http://news.whu.edu.cn/_mediafile/whu_news/2015/04/14/2ppgcrb9dk.jpg" />
```

相对 URL 用于引用相对于使用 URL 位置的另一个资源,所以当前网页的位置很重要。例如,若当前网页位于网站的根目录,该网页中的如下代码可以引用网站的 Images 目录中的 img4. bmp 文件:

```
<img src="Images/img4.bmp " alt="图像" />
```

若当前网页位于网站根目录的 Student 子目录中,该网页中的如下代码可以引用网站的 Images 目录中的 img4. bmp 文件:

```
<img src="../Images/img4.bmp " alt="图像" />
```

其中,开头的两个点用于导航到网站根目录。

在服务器端,可以用波浪符号指向网站根目录。例如,该网站的任何网页中的如下代码都可以引用网站的 Images 目录中的 img4. bmp 文件:

```
<img  src="~/Images/img4.bmp" />
```

【练一练】 在 CH4 网站中添加一个 webform8. html 网页,其功能是说明图像标记的应用。

其设计步骤如下:

① 打开 CH4 网站,选择"网站|添加新项"命令,出现"添加新项-CH4"对话框,在中间列表中选择"HTML 页",将文件名称改为 webform8. html,单击"添加"按钮。

② 在解决方案资源管理器中,右击网站名 CH4,在出现的快捷菜单中选择"添加|新建目录"命令,添加一个名称为 Images 的目录,并放入几个图像文件。

③ 进入源视图,设计其 HTML 代码如下:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
      <title></title>
    </head>
    <body>
      <table>
          <tr>
              <td>手机</td>
              <td>相机</td>
          </tr>
          <tr>
              <td><img src="Images/1122.jpg" alt="图像 1" /></td>
              <td><img src="Images/1221.jpg" alt="图像 2" /></td>
```

```
            </tr>
            <tr>
                <td>笔记本</td>
                <td>台式机</td>
            </tr>
            <tr>
                <td><img src="Images/2121.jpg" alt="图像 3" /></td>
                <td><img src="Images/2211.jpg" alt="图像 4" /></td>
            </tr>
        </table>
    </body>
</html>
```

该网页中有一个 4 行 2 列的表格，第 1、3 行的各列显示商品名称的文字，第 2、4 行的各列显示商品对应的图片。该网页在 IE 浏览器中的显示结果如图 4.11 所示。



图 4.11　webform8.html 网页的执行界面

4.2.8　HTML 框架标记

框架也是网页布局的重要工具，它与表格的不同之处在于表格是把网页分割成小的单元格，而框架是把浏览器的窗口分割成若干个小窗口，这些子窗口称为框架，每一个框架都相当于一个浏览器窗口，这样就使一个浏览器窗口可以显示多个网页。HTML 主要的框架标记及其说明如表 4.7 所示。

表 4.7　HTML 主要的框架标记及其说明

框架标记	功能
<frame>	定义框架集的窗口或框架
<frameset>	定义框架集
<noframes>	定义针对不支持框架的用户的替代内容
<iframe>	定义内联框架

1. 建立框架

基本用法：

```
<frameset 属性="属性值">…</frameset>
```

其功能是指定当前窗口的分割结构，其分为几行还是分为几列。框架标记常用的属性如下：

- rows="高度列表"：设置子窗口的高度，即把整个窗口横向分割成几个框架(垂直框架)。
- cols ="宽度列表"：设置子窗口的宽度，即把整个窗口纵向分割成几个框架(水平框架)。

其中，rows 指明当前窗口要分为几行及各行的高度；cols 用于指定列。例如，要把当前窗口分成等高的两行，可以按如下这样：

```
<frameset rows="50%,50%">…</frameset>
```

“高度列表”(或“宽度列表”)中数字个数表示要分割的行(或列)数，各数字的大小表示相应行(或列)的高度(或宽度)，其对应顺序为从上到下(或从左到右)。各数字的取值可以为具体整数值(其单位为像素)，可以为当前窗口高度(或宽度)的百分数。其中，在列表数字中可以有一个数字被指定为 *。这个 * 表示相应的行高(或列宽)为指定了其他行(或列)的高度(或宽度)后当前窗口剩余的高度(或宽度)。

注意：设计网页时，不能将<frameset></frameset>标记放置在<body></body>标记内部。另外，在同一个<frameset>中不能既指定 rows 又指定 cols，因为没有足够的信息提供给浏览器通知它在分行后应该在那一行中分列。

2. frame 标记

一个 frame 表示一个窗口，它嵌入在<frameset>…</frameset>之间，按照 rows 和 cols 设定的子窗口顺序，依次指定一个子窗口显示哪一个网页。其基本用法如下：

```
<frame 属性="属性值" … />
```

其常用的属性如下：

- src="url"：设置要链接到该子窗口的 URL。
- name="framename"：表示子窗口的名称。
- marginwidth="size"：用来控制显示内容和窗口左右边界的距离，默认为 1。
- marginheight= "size"：用来控制显示内容和窗口上下边界的距离，默认为 1。
- scrolling="yes/no/auto"：指定子窗口是否使用滚动条，默认 auto，即根据窗口内容决定是否有滚动条。
- noresize：使用该属性后，指定窗口不能调整窗口大小。

例如，如下代码使用三份不同的文档制作一个如图 4.12 所示的水平框架(其中 HtmlPagea.html 等 3 个网页分别在浏览器中显示“框架 A”、“框架 B”和“框架 C”文字)：

```
<frameset cols="30%,40%,30%">
    <frame src="HtmlPagea.html" />
    <frame src="HtmlPageb.html" />
    <frame src="HtmlPagec.html" />
</frameset>
```

而如下代码使用三份不同的文档制作一个如图 4.13 所示的垂直框架：

```
<frameset rows = "30 % ,40 % ,30 % ">
    <frame src = "HtmlPagea.html" />
    <frame src = "HtmlPageb.html" />
    <frame src = "HtmlPagec.html" />
</frameset>
```

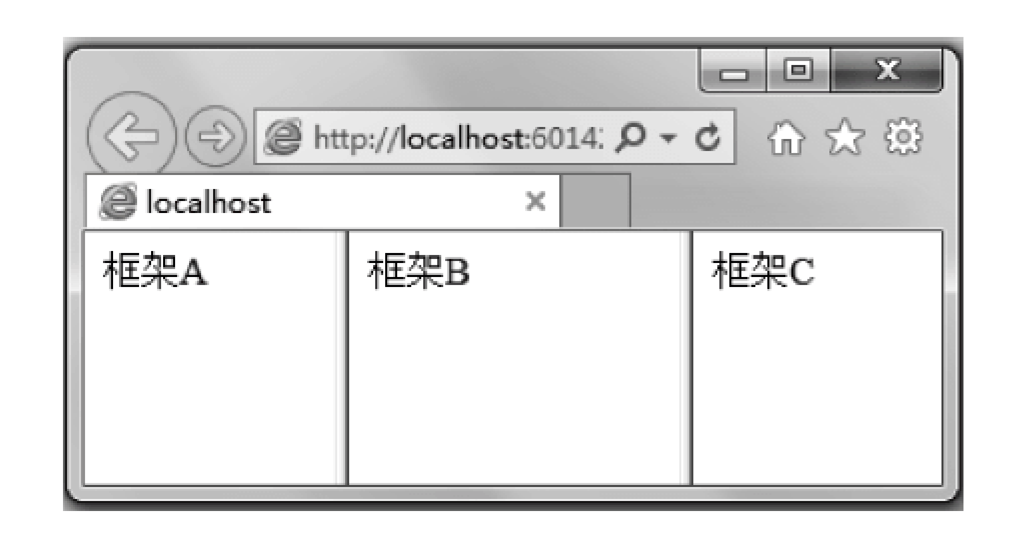


图 4.12　一个水平框架



图 4.13　一个垂直框架

3. iframe 标记

iframe 标记用于创建包含另外一个文档的内联框架(即行内框架)。其基本用法如下:

```
< iframe 属性 = "属性值"> … </iframe>
```

iframe 框架的常用属性如下。

- src="url":设置要链接到该框架的 URL。
- width="size":设置 iframe 框架的宽度。
- height="size":设置 iframe 框架的高度。
- name="name":设置 iframe 框架的名称。
- frameborder="size":指定 iframe 框架是否有边框,size 可取 1 和 0 值之一,默认值为 1。
- marginwidth="size":用来控制显示内容和窗口左右边界的距离,默认值为 1。
- marginheight= "size":用来控制显示内容和窗口上下边界的距离,默认值为 1。
- scrolling="yes/no/auto":指定子窗口是否使用滚动条,默认值为 auto,即根据窗口内容决定是否有滚动条。

由于 iframe 框架可以显示任意一个网页,在本书后面的电子商务综合实例中大量使用到 iframe 框架。

【练一练】 在 CH4 网站中添加一个 webform9.html 网页,采用 iframe 框架标记实现网页导航功能。

其设计步骤如下:

① 打开 CH4 网站,选择"网站|添加新项"命令,出现"添加新项-CH4"对话框,在中间列表中选择"HTML 页",将文件名称改为 webform9.html,单击"添加"按钮。

② 进入源视图,设计其 HTML 代码如下:

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
```

```
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
        <title></title>
    </head>
    <body>
          <table>
             <tr>
               <td>
                  <h2>导航</h2>
                  <a href ="webform1.html" target ="showframe">webform1</a>
                  <br />
                  <a href ="webform2.html" target ="showframe">webform2</a>
                  <br />
                  <a href ="webform3.html" target ="showframe">webform3</a>
               </td>
               <td>
                   <iframe name="showframe"></iframe>
              </td>
           </tr>
        </table>
    </body>
</html>
```

该网页中有一个 1 行 2 列的表格，第 1 列显示 3 个超链接，第 2 行有一个 iframe 框架(名称为 showframe)。运行该网页如图 4.14 所示。用户单击任何超链接，则在 showframe 框架中显示个网页。用户单击 webform2 超链接的显示结果如图 4.15 所示。

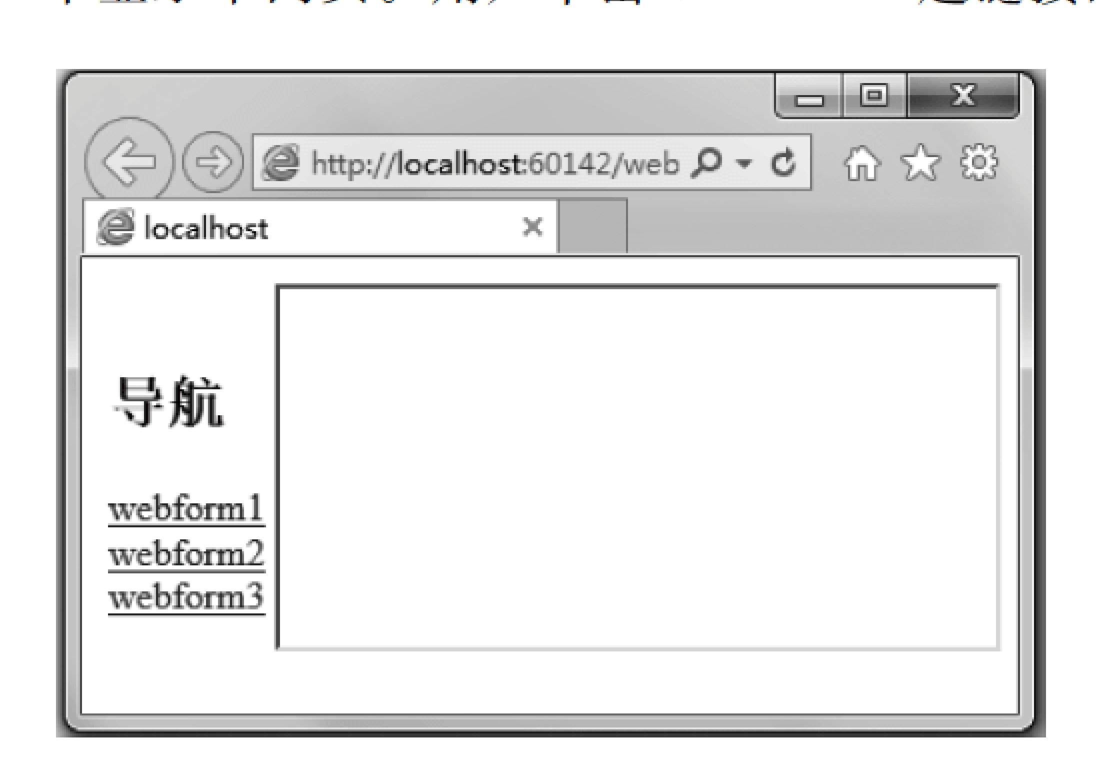




图 4.14 webform2.html 网页的执行界面一



图 4.15 webform2.html 网页的执行界面二

从中看到，frameset/frame 和 iframe 框架的几点差别如下：

- HTML5 不支持 frameset/frame，但支持 iframe。
- frame 不能脱离 frameset 单独使用，而 iframe 可以。
- frame 的高度只能通过 frameset 控制，而 iframe 可以自己控制。
- frame 不能放在 body 中，而 iframe 可以。
- iframe 的使用更加随意和灵活。

4.2.9 HTML 表单标记

HTML 提供的表单(form)是用来将用户输入的数据从浏览器传递给 Web 服务器的。例如，可以利用表单建立一个用户注册界面，也可以利用表单对数据库进行查询。表单的主要标记如表 4.8 所示。

表 4.8　主要的表单标记及其说明

表单标记	功　能
<form>	定义供用户输入的 HTML 表单
<input>	定义输入控件
<textarea>	定义多行的文本输入控件(文本域)
<button>	定义按钮
<select>	定义选择列表(下拉列表)
<optgroup>	定义选择列表中相关选项的组合
<option>	定义选择列表中的选项
<label>	定义 input 元素的标注
<fieldset>	定义围绕表单中元素的边框
<legend>	定义 fieldset 元素的标题
<datalist>	定义下拉列表
<keygen>	定义生成密钥
<output>	定义输出的一些类型

1. <form>标记

基本用法：

```
<form 属性 = "属性值"> … </form>
```

<form>标记的主要属性如下：

- action(必选)：用来指出当这个表单提交后需要执行的驻留在 Web 服务器上的程序名(包括路径)是什么。一旦 Internet 网络用户提交输入信息后服务器便激活这个程序，完成某种任务。例如，<form action="login.aspx" method="post">…</form>，当用户单击本表单的提交按钮以后，Web 服务器上的 login.aspx 将接收用户输入的信息，以登记用户信息。
- method(可选)：用来说明从客户端浏览器将因特网用户输入的信息传送给 Web 服务器时所使用的方式，它有 post 和 get(默认方式)两种方式，前者从指定的资源请求数据，后者向指定的资源提交要被处理的数据。从数据的可见性看两者的区别是，使用 post 时表单中所有的变量及其值按规律放入报文中，而不是附加在 action 所设定的 URL 之后；使用 get 时将 form 的输入信息作为字符串附加在 action 所设定的 URL 的后面，并用？隔开，即在客户端浏览器的地址栏中可以直接看见这些内容。
- enctype(可选)：该属性规定在发送到服务器之前应该如何对表单数据进行编码。默认地，表单数据会编码为 application/x-www-form-urlencoded。在发送到服务器之前，所有字符都会进行编码(空格转换为“+”加号，特殊符号转换为 ASCII 十六进制值)。
- target(可选)：用于规定在哪一个窗口中打开 action 属性中规定的网页，默认值为 _self。

2. 各种常见的表单控件

在<form>与</form>之间可以嵌入各种控件，也称为表单域标记。它们的通用格式为：

```
<input type = "输入控件类型" name = "域名称" value = "值">
```

其中,type属性设置该控件的类型;name确定该控件在整个文档中的名称;value属性设置input元素的值。

(1) 单行文本输入框

基本用法:

```
< input type = "text" name = "域名称" value = "默认值" maxlength = 值 size = 值 />
```

其中,value属性确定该文本框预置的文字;maxlength属性确定在这个文本框中所能容纳的字符串最大长度,该项可以不设置;size属性确定这个文本框的显示宽度,以能显示多少个字符来衡量。

另有一种特殊的单行文本输入框专门用于输入密码(password),不同之处在于它对键盘输入的回显字符为*,即它把用户的输入隐藏了。其用法为:

```
< input type = "password" name = "域名称" value = "默认值" maxlength = 值 size = 值>
```

(2) 命令按钮

基本用法:

```
< input type =  "button" name = "域名称" value = "值">
```

type设置为"button"表示这个控件为按钮。另有两个特殊的控件,它们实质上是按钮,但其type不是"button",而分别是"submit"(提交按钮)和"reset"(重置按钮)。其用法分别为:

```
< input type = "submit" name = "域名称" value = "值">
< input type = "reset" name = "域名称" value = "值">
```

按下提交按钮后,表单就把当前所获得的信息以method指定的方式全部传给action指定的程序。按下重置按钮后,则表单中的所有控件都被重置,恢复初始状态。

(3) 复选框

基本用法:

```
< input type = "checkbox" name = "域名称" value = "值" checked >
```

type设置为"checkbox"表示这个控件为复选框;value用于设置当这个检查框被选中后,发送给action指定处理程序的值;checked为预置该检查框被选中,如果有这一项,该检查框初始值为被选中。

(4) 单选框(选项按钮)

基本用法:

```
< input type = "radio" name =  "域名称" value = "值" checked >
```

单选框的各属性意义与复选框的基本相同。值得注意的是,要设置单选框时,各选项必须同名(具有相同的name),取值不同(value不相同),并且几个选项中必须有且只能有一个预置为选中。如果没有预置选中项,默认预置第一个选择项被选中。

(5) 图像

基本用法:

```
< input type = "image" name = "域名称" src = URL >
```

该标记把 src 指定位置的图像加到表单里，当用户用鼠标在该图像内点击时，该点在图像中的坐标作为值传给 action 指定的处理程序。

(6) 隐藏项

基本用法：

```
< input type = "hidden" name = "域名称" value = 值>
```

该控件的内容在表单中被隐藏起来，并不在网页中显示。通常可用来以隐藏方式向服务器传送有关信息。

(7) 菜单和列表

select 元素可创建单选或多选菜单。当提交表单时，浏览器会提交选定的项目，或收集用逗号分隔的多个选项，将其合成一个单独的参数列表，并且在将<select>表单数据提交给服务器时包括 name 属性。

option 元素定义下拉列表中的一个选项，浏览器将<option>标签中的内容作为<select>标签的菜单或是滚动列表中的一个元素显示。option 元素位于 select 元素内部。

select 元素和 option 元素的基本用法如下：

```
< select >
  < option value  = "选项 1 名称">选项 1 显示文字</option >
  < option value  = "选项 2 名称">选项 2 显示文字</option >
  ⋮
  < option value  = "选项 n 名称">选项 n 显示文字</option >
</select >
```

【练一练】 在 CH4 网站中添加一个 webform10.html 网页，通过一个用户注册界面的设计说明表单标记的使用方法。

其设计步骤如下：

① 打开 CH4 网站，选择"网站|添加新项"命令，出现"添加新项-CH4"对话框，在中间列表中选择"HTML 页"，将文件名称改为 webform10.html，单击"添加"按钮。

② 进入源视图，设计其 HTML 代码如下：

```
<! DOCTYPE html >
< html xmlns = "http://www.w3.org/1999/xhtml">
  < head >
    < meta http - equiv = "Content - Type" content = "text/html; charset = utf - 8"/>
    < title > </title >
  </head >
  < body >
    < form action = "RegPage.html" method = "post">
       < table style = "align - content:center;width:350px">
          < caption >< h3 >用户注册</h3 ></caption >
          < tr >
            < td style = "text - align:right">用户名</td >
            < td > < input type = "text" name = "name" size = "20"
                     style = "width: 120px; height: 16px" />
            </td >
          </tr >
          < tr >
             < td style = "text - align:right">密码</td >
             < td >
```

```
                <input type="password" name="pass" size="20"
                    style="width: 120px; height: 16px" />
            </td>
        </tr>
        <tr>
          <td style="text-align:right">年龄</td>
          <td><input type="radio" name="age" />18 以下
                <input type="radio" name="age" />18～25
                <br />
                <input type="radio" name="age" />26～45
                <input type="radio" name="age" />45 以上
          </td>
        </tr>
        <tr>
          <td style="text-align:right">民族</td>
          <td>
              <select>
                <option value  ="mz1">汉族</option>
                <option value  ="mz2">满族</option>
                <option value  = "mz3">回族</option>
                <option value  = "mz4">其他</option>
              </select>
            </td>
        </tr>
        <tr>
          <td style="text-align:right">喜爱的运动</td>
          <td><input type="checkbox" name="love1" value="足球"/>足球
              <br />
              <input type="checkbox" name="love2" value="篮球" checked="checked" />篮球
              <br /><input type="checkbox"  name="love3" value="排球"/>排球
          </td>
        </tr>
        <tr>
          <td style="text-align:center" colspan="2">
                <input type="submit" value="确定" />   
                <input type="reset" value="重置" />
          </td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

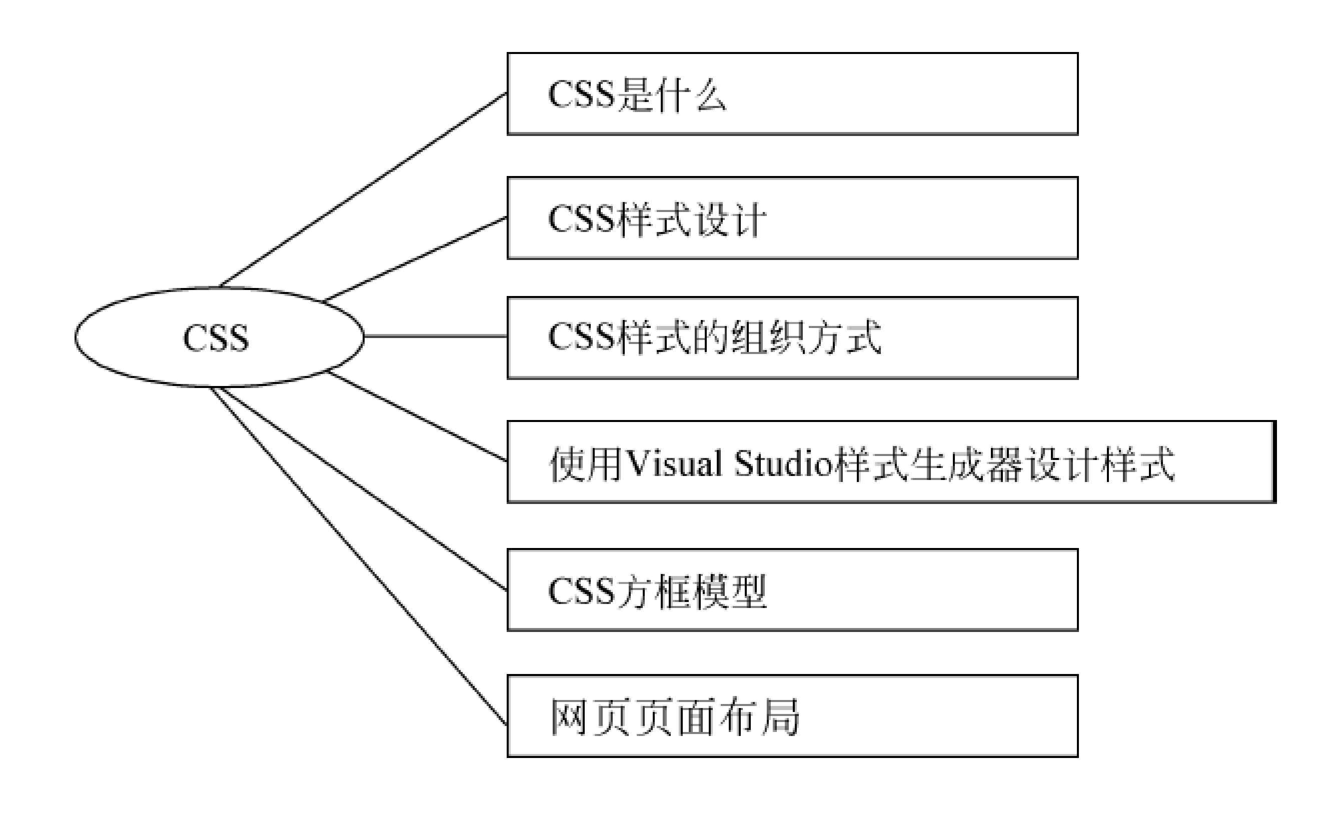
该网页中包含一个 6 行 2 列的表格，并设计了表格标题。表格中使用单行文本输入框、命令按钮、复选框、单选框、菜单和列表等表单控件。该网页在 IE 浏览器中显示的初始界面如图 4.16 所示。用户在其中操作后单击“确定”按钮，将转向 RegPage.html 网页（这里没有设计 RegPage.html 网页）。



图 4.16 webform10.html 网页的执行界面

4.3　CSS 设计

知识梳理



4.3.1　CSS 是什么

CSS(Cascading Style Sheets)即为级联样式表或层叠样式表，与 HTML 一样也是一种标记语言，甚至很多属性都是来源于 HTML。CSS 技术诞生于 1996 年，由 W3C 负责组织和制定的。

CSS 与 HTML 的关系就是“内容”和“形式”，即 HTML 来组织网页的结构和内容，CSS 来决定页面的表现形式。CSS 的优点如下：

- 和传统的 HTML 相比，CSS 除了具有强大的控制能力和排版能力之外，最主要的是实现了内容与样式的分离，这种做法带来了许多好处；
- CSS 简化了网页的代码，提高了访问速度；
- 采用 CSS 可以构建公共样式库，便于重用样式；
- CSS 便于修改网站的样式；
- CSS 方便团队的开发。

4.3.2　CSS 样式设计

样式是指每一个网页元素呈现在浏览器中的风格，如字体的大小、颜色、页面的背景色、背景图等。样式表中包含应用于网页元素的相关样式信息。

定义样式的基本格式如下：

```
样式属性 1:值 1; 样式属性 2:值 2; …
```

样式属性与值之间用冒号“:”分隔，如果一个样式中有多个样式属性，各样式属性之间要用分号“;”隔开。

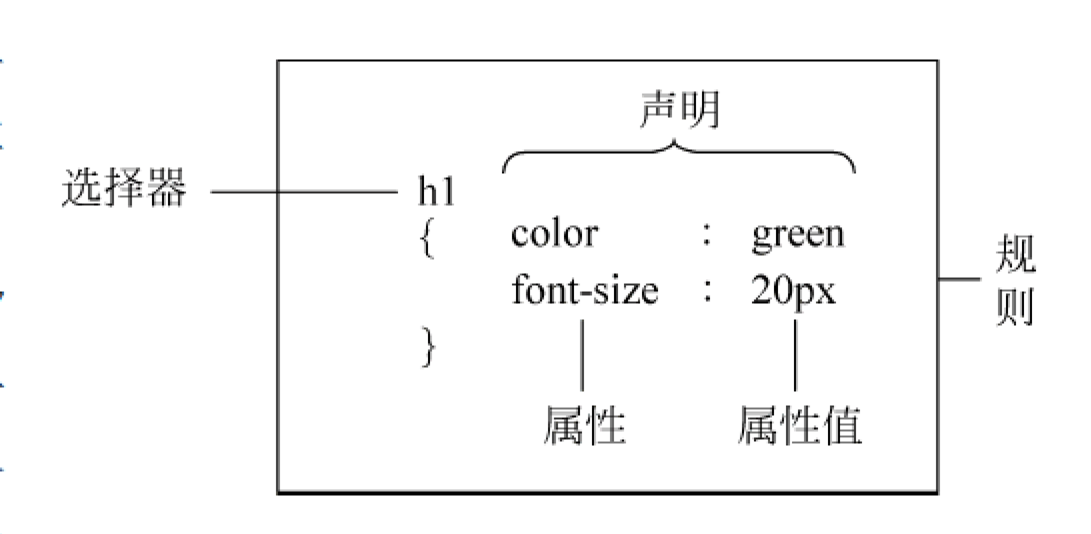


图 4.17　最简单样式表的形式

例如，一个最简单样式表的形式如图 4.17 所示。其中，h1 称为选择器，用来表示应当向什么元素应用该格式化信息。从 h1 开始到闭合大括号的代码块称为规则。上述规则定义了网

页中所有<h1>元素的外观。该选择器可以直接映射到 HTML 元素上。

一般地，要能够样式化网页上的元素，浏览器必须知道 3 个事件：

- 必须样式化网页上的什么元素？
- 必须样式化元素的什么部分？
- 希望选中的元素的那部分看起来是什么样子？

这些问题的答案由选择器、属性和值给出。

1. CSS 选择器

在 CSS 中，选择器是一种模式，用于选择需要添加样式的元素。CSS 中的选择器有 Universal 选择器、Type 选择器、类选择器、ID 选择器和伪类选择器等。

（1） Universal 选择器

Universal 选择器(通用选择器)用星号(*)表示，适用于网页的所有元素。Universal 选择器可以用来进行一些全局设置。例如，以下规则将网页中所有元素的字体改为 Arial：

```
* { font - family : Arial; }
```

（2） Type 选择器

Type 选择器(类型选择器)是最典型的选择器类型，用于指向一个特定类型的 HTML 元素。如果有多个不同的标记要使用相同的样式，则可以采用编组的方法简化定义。例如：

```
h1,h2,h3 { color:red }
```

则所有的 h1、h2、h3 标题都将以红色显示，这种表示法中，各选择器之间要用逗号“,”分隔。

（3） 类选择器

使用类选择器可以为某一个 HTML 标记创建多个样式，或为多个标记创建同一种样式。类选择器的定义格式如下：

```
样式定义选择符.类名{样式属性 1:值 1;样式属性 2:值 2;… }
```

例如，h1. first 和 h1. second 的样式代码分别如下：

```
h1.first {color:red;font - size:40px }
h1.second{color:blue;font - size:30px }
```

这样，可以通过以下方式使用它们：

```
<body>
   <h1 class = "first">中华人民共和国</h1>
   <h1 class = "second">教育部</h1>
</body>
```

其在浏览器中的显示结果如图 4. 18 所示。

（4） ID 选择器

选择器用来在网页内选择或指向一个或多个特定的元素，以对要格式化的元素更细化的控制。可以使用若干个不同的选择器，这里仅介绍 ID 选择器。

ID 选择器以“#”为标志，依靠这个唯一的标志可以定义一套样式。其定义方法如下：

中华人民共和国

教育部

图 4.18　采用类选择器样式的浏览器显示结果

```
#ID 选择器名 {属性 1:值 1;属性 2:值 2; …; 属性 n:值 n }
```

其中，ID 选择器名前的“#”符号一定不能省略。例如：

```
#customId1 {color:red}
```

在网页中引用该样式的标记内使用 id 属性即可。例如：

```
<p id = "customId1">本段落文字为红色</p>
```

ID 选择器与类选择器主要区别如下：

- 类选择器前面以“.”开始，而 ID 选择器前面以“#”开始。
- 在设计网页时，类可以分配给任何个数的元素，通常 ID 选择器只能在某个 HTML 文档中使用一次。ID 选择器类似于表单元素 input 中的 name 属性，每个 name 属性的值应该是唯一的。
- 值得注意的是，实际上有些浏览器(如 IE)不一定会检查网页中 ID 选择器的唯一性，可以在网页中对多个元素使用同一个 ID 选择器，从而使同一个样式表现在多个元素上。建议最好不要这要做。
- ID 选择器对元素应用样式时比类选择器具有更高的优先级。

(5) 伪类选择器

CSS 还包含一系列伪类选择器(简称为伪类)，在创建样式规则时提供了额外的选项。伪类可以添加到其他选择器以创建更复杂的 CSS 规则。

例如，有如下代码：

```
#title p:first-child  { font-size: small; color:red; }
#title p:nth-child(2) { font-size: medium; color:blue;}
```

第 1 行表示将该样式应用于 id 属性为 title 元素中的第一个段落标记。第 2 行表示将该样式应用于 id 属性为 title 的元素中的第 2 个段落标记。例如：

```
<body>
    <div id = "title">
        <p>第 1 个段落: 红色 small</p>
        <p>第 2 个段落: 蓝色 medium</p>
    </div>
</body>
```

其在浏览器中的显示结果如图 4.19 所示。

第1个段落：红色small

第2个段落：蓝色medium

图 4.19　采用伪类样式的浏览器显示结果

另外，还有与锚点相关的伪类，是专用于<a>标记的选择器，可以设置不同类型超链接的显示方式：

- a:link：未被访问过的超链接。
- a:visited：已被访问过的超链接。
- a:active：当超链接处于选中状态。
- a:hover：当鼠标指针移动到超链接上。

例如：

```
a:visited,a:link { color:blue }
a:hover { color:red; text-decoration:none }
```

语句定义了这个文档中的超链接文本在未访问和被访问时为蓝色、带下划线，当有鼠标掠过时颜色变为红色、不带下划线。

2. CSS 属性

属性是元素的一部分，可通过样式表修改。CSS 定义了一个很长的属性列表，如 CSS 文本属性(text)、CSS 背景属性(background)、CSS 尺寸属性(dimension)和 CSS 表格属性(table)等。大多数情况下网站中不会用到所有项，表 4.9 给出了常用的 CSS 属性。

表 4.9　常用的 CSS 属性

CSS 属性	功　能
background-color	设置元素的背景颜色
background-image	设置元素的背景图像
border	在一个声明中设置所有的边框属性
height	设置元素高度
width	设置元素的宽度
font	在一个声明中设置所有字体属性
font-family	规定文本的字体系列
font-size	规定文本的字体尺寸
font-weight	规定字体的粗细
target	简写属性，设置 target-name、target-new 以及 target-position 属性
margin	在一个声明中设置所有外边距属性
padding	在一个声明中设置所有内边距属性
color	设置文本的颜色
text-align	设置文本的水平对齐方式
vertical-align	设置元素的纵向排列对齐方式

另外还有两个很重要的定位属性。

(1) float 属性

该属性定义元素在哪个方向浮动。以往这个属性总应用于图像，使文本围绕在图像周围，不过在 CSS 中，任何元素都可以浮动。浮动元素会生成一个块级框，而不论它本身是何种元素。该属性可能的取值如下：

- none(默认值)：元素不浮动，并会显示在其在文本中出现的位置。
- left：元素向左浮动。
- right：元素向右浮动。
- inherit：规定应该从父元素继承 float 属性的值。

(2) position 属性

该属性规定元素的定位类型，定义建立元素布局所用的定位机制。任何元素都可以定位，不过绝对或固定元素会生成一个块级框，而不论该元素本身是什么类型。相对定位元素会相对于它在正常流中的默认位置偏移。可能的取值如下：

- static(默认值)：没有定位，元素出现在正常的流中(忽略 top、bottom、left 或 right 等声明)。
- absolute：生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。元素的位置通过 left、top、right 或 bottom 属性进行规定。
- fixed：生成绝对定位的元素，相对于浏览器窗口进行定位。元素的位置通过 left、top、

right 或 bottom 属性进行规定。

- relative：生成相对定位的元素，相对于其正常位置进行定位。因此，left:20px 表示向元素的左边位置添加 20 像素。
- inherit：规定应该从父元素继承 position 属性的值。

例如，如下代码采用绝对定位和相对定位显示文字：

```
<body>
    <div style="position:absolute;left:40px;top:60px">
        <h2>绝对定位的标题</h2>
    </div>
    <div style="position:relative: left:20px">
        <h3>相对定位的标题</h3>
    </div>
</body>
```

其显示结果如图 4.20 所示。

实际上，Visual Studio 提供了十分方便的智能感知，当开发人员输入 CSS 属性名时，智能感知便列出相应的 CSS 属性，如图 4.21 所示，可以从中选择相应 CSS 属性。因此不必全部记住这些 CSS 属性。



图 4.20　采用绝对定位和相对定位显示文字的结果

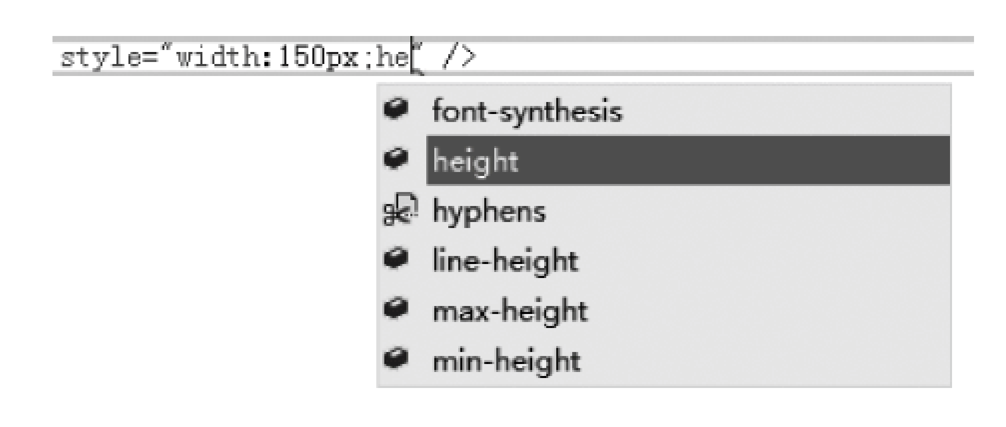


图 4.21　智能感知显示的 CSS 属性一

3. 属性值

从前面 float 和 position 属性看到，同一属性的不同取值会导致不同的效果。

另外，与属性一样，值也有很多风格。可用的值取决于具体的属性。例如，color 属性采用表示颜色的值，可以是颜色名称(如 red)，也可能是代表红、绿、蓝色成分的十六进制数(如 #FF0000)。

同样，当开发人员输入完一个 CSS 属性名时，智能感知便列出相应的 CSS 属性值，如图 4.22 所示列出的是 color 的 CSS 属性值，可以从中选择相应值。

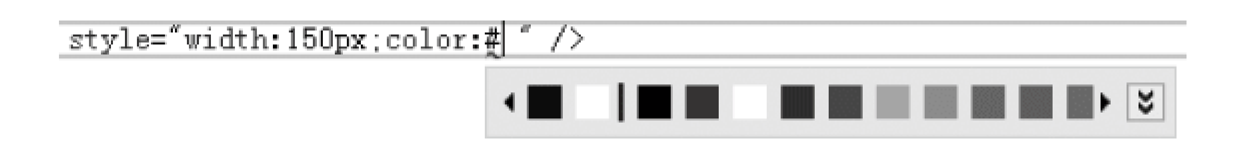


图 4.22　智能感知显示的 CSS 属性值二

4.3.3　CSS 样式的组织方式

样式的组织方式主要有 3 种，即内联样式、内部样式表和外部样式表。无论 CSS 样式如何组织，一旦服务器将它们发送到客户端，浏览器将负责解析样式，把它们应用于网页中相应

的 HTML 元素。而当采用内部样式表或外部样式表时，样式被定义为 CSS 规则，浏览器使用该规则确定应用什么样式，以及应用于哪些 HTML 元素。

1. 内联样式

在网页设计中，大多数 HTML 元素都有 style 属性，每个 HTML 元素使用<style>标记建立一个或多个样式，这种方式就是内联样式方式，也称为网页内嵌法。其用法如下：

```
style = "属性 1:值 1; 属性 2:值 2; …; 属性 n:值 n"
```

内联样式不需要定义为规则，因为它们会自动应用于包含它们的元素。因此，浏览器不需要选择要应用该样式的元素。例如：

```
<h1 style = "font - size:40px;color:Red;">中华人民共和国</h1>
<h2 style = "font - size:30px;color:Blue;">教育部</h2>
                 内联样式
```

这样，浏览器直接将 style 属性指定的样式作用于各自的元素。

这种方式的优点是直观、方便；缺点是不喜欢某种样式需要不厌其烦的重新逐一修改每一个元素的样式。

2. 内部样式表

设计一个网页可能需要多个样式，内部样式表方式就是将是在单个网页中用到的样式集中存储在该网页内部。内部样式表就是单个网页内部存储的 CSS 样式集合（也称为私有样式）。这些样式位于<style>标记中，这个标记一般位于网页的<header>部分。

例如（粗体部分为 style 标记）：

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
     <meta http - equiv = "Content - Type" content = "text/html; charset = utf - 8"/>
     <title>样式引用示例</title>
     <style type = "text/css">
          h1 {font - size:40px; color:Red;}      内部样式表
          h2 {font - size:30px; color:Blue;}
     </style>
  </head>
  <body>
     <h1>中华人民共和国</h1>
     <h2>教育部</h2>
  </body>
</html>
```

这种方法的优点是所有样式集中放在一起，便于修改，一旦某个样式发生改变，本网页中所有该样式的元素都发生更改。

3. 外部样式表

前面两种方式设计的样式都只适用于它所在的网页。如果要将其用于其他网页，最好把所设计的样式放在一个独立的文件中，这样的文件就是外部样式表文件。例如，样式表文件 StyleSheet1. css 的内容为：

```
body { background - color:#33bb66; }
```

```
h1 { font - size:40pt; color:Blue; }
h2 { font - size:30pt; color:White;}
```

在网页文件中引用该样式表文件只需要在网页的<head>元素中添加如下代码：

```
< link href = "StyleSheet1.css" type = "text/css" rel = "Stylesheet" />
```

其中，rel 规定了被链接文件的关系，取值是“Stylesheet”；type 属性规定了链接文件的类型；href 属性则指定了要链接的样式表文件的 URL。

凡是在网页的<head>元素中与该样式表文件建立链接的 HTML 文件，其网页元素的样式就会按照外部样式表文件中的定义显示。

外部样式表文件的扩展名为 css。在 Visual Studio 中，创建外部样式表文件的操作是：选择“网站|添加新项”菜单命令，出现“添加新项”对话框，在中间列表中选择“样式表”，设置样式表文件名，单击“添加”按钮。

说明：在 3 种样式表组织方式中，一般地，外部样式表优于内部样式表，而内部样式表优于内联样式。但在实际应用中，究竟采用哪种方式需要根据具体应用而定。

4.3.4　使用 Visual Studio 样式生成器设计样式

Visual Studio 提供了专门的样式生成器可以可视化地设计样式表文件。

【练一练】　在 CH4 网站中添加一个 webform11.html 网页，并添加一个 StyleSheet.css 外部样式表文件，设计相应的样式，并将其应用到网页设计中。

其设计步骤如下：

① 打开 CH4 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH4”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform11.html，单击“添加”按钮。

② 选择“网站|添加新项”菜单命令，出现“添加新项-CH4”对话框，在中间列表中选择“样式表”，保持默认文件名称为 StyleSheet.css，单击“添加”按钮。

③ 出现如图 4.23 所示的样式设计对话框。在样式编辑窗口删除原有内容，输入“h1 { }”，将光标移动到该大括号内，单击工具栏的 按钮（或在该大括号内右击鼠标，在出现的快捷菜单中选择“生成样式”命令），出现“修改样式”对话框。

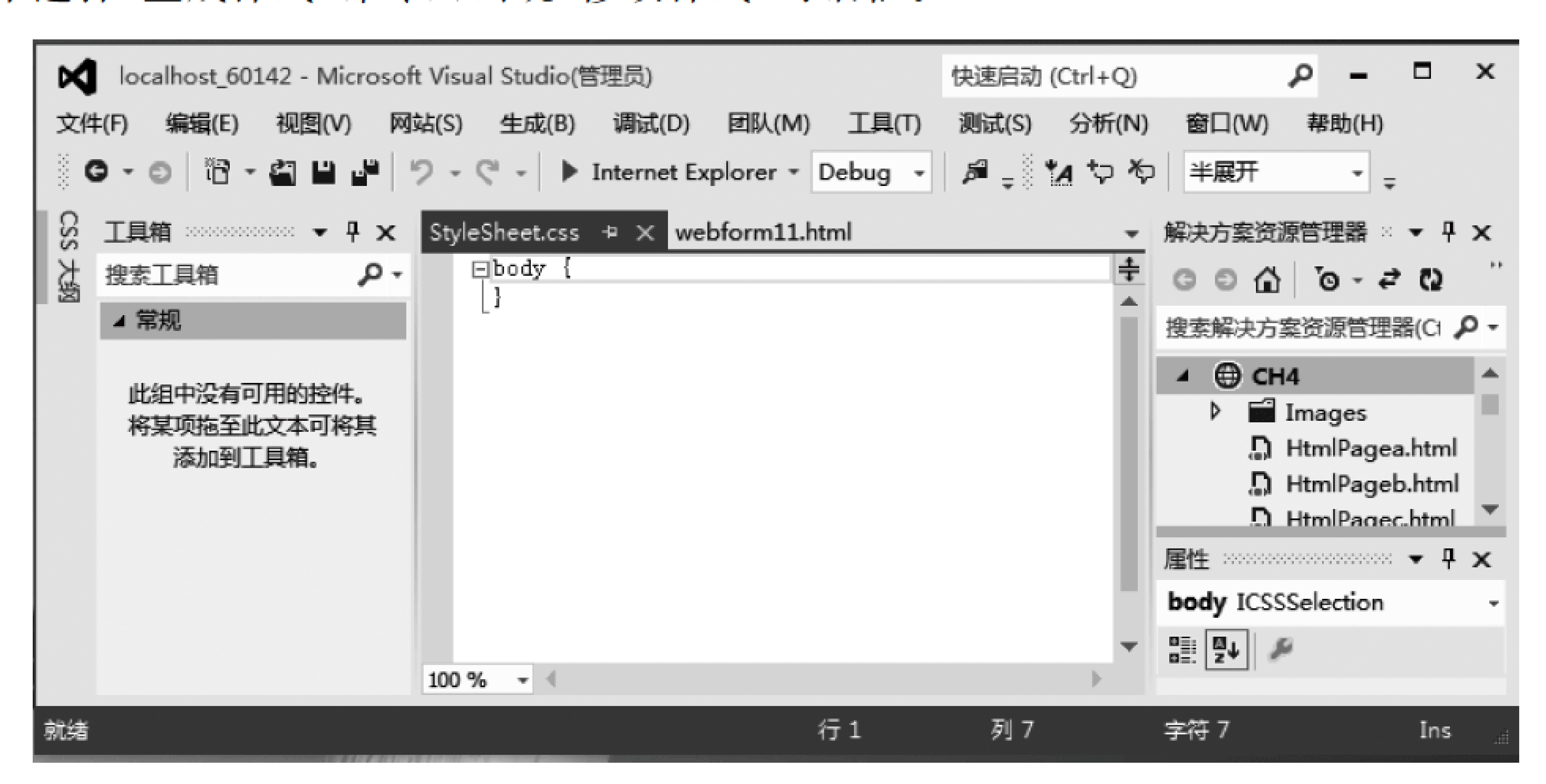


图 4.23　样式设计对话框

④ 在“修改样式”对话框中设置 h1 的“字体”类别如图 4.24 所示，单击“确定”按钮返回。此时，h1 的定义变为：

```
h1 {  color: #FF0000;     font-weight: bold;
      font-size: larger;  font-family: 楷体;
}
```



图 4.24　“修改样式”对话框

⑤ 采用同样的操作创建两个类选择器如下：

```
h1.first  { font-size: 35px; color: #0000FF;  }
h1.second { font-size: 25px; color: #006600; }
```

单击工具栏的 按钮保存该外部样式表文件。

⑥ 切换到 webform11.html 网页，进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <link href="StyleSheet.css" type="text/css" rel="Stylesheet" />
  </head>
  <body>
    <div>
          <h1>中华人民共和国</h1>
          <h1 class="first">中华人民共和国</h1>
          <h1 class="second">中华人民共和国</h1>
    </div>
  </body>
</html>
```

该网页在 IE 浏览器中的显示结果如图 4.25 所示。从中看到，第 1 行文字应用了 h1 样式，第 2 行文字应用了 h1.first 类样式，第 3 行文字应用了 h1.second 类样式，且后两行都应

用了 h1 的加粗和楷体。

该样式表文件可以被本网站中的多个网页所引用，达到样式共享的目的，而且使整个网站界面具有一致性。

在网页设计中，选择“视图|管理样式”菜单命令，会出现“管理样式”对话框，如图 4.26 所示是 webform11.html 网页的“管理样式”对话框，通过该对话框可以查看当前网页上能使用的所有样式。其中，按钮用于新建样式；按钮用于附加样式表。



图 4.25　webform12.html 网页的执行界面



图 4.26　“管理样式”对话框

4.3.5　CSS 方框模型

在 CSS 中，方框模型是定位元素的核心，定义了浏览器将 HTML 中的每个元素看作矩形方框。该方框由不同的部分组成，包括页边距、内边距边框和内容，如图 4.27 所示。

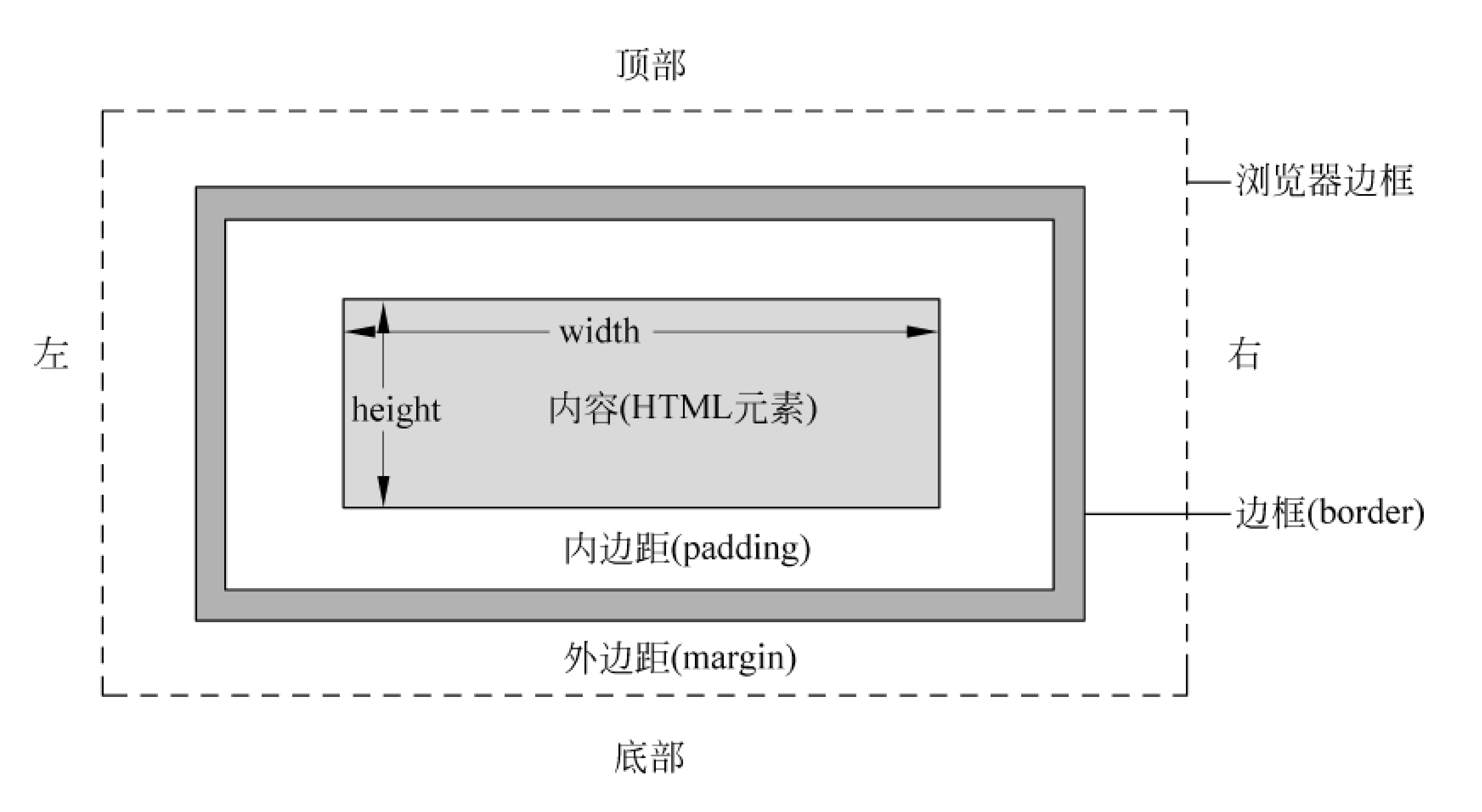


图 4.27　CSS 方框结构

1. 设置方框的外边距

CSS 的外边距属性设置为：

```
margin: margin - top margin - right margin - bottom margin - left
```

其中，margin 设置对象与浏览器边框之间的边距；margin-left 设置左边距；margin-right 设置右边距；margin-top 设置上边距；margin-bottom 设置下边距。可以使用其中任何一个属性

来设置相应的外边距，而不会直接影响其他外边距。

例如，以下代码指定段落的左外边距为 2cm：

```
<style type="text/css">
    p.leftmargin {margin-left: 2cm}
</style>
```

2. 设置方框的边框

每个边框有宽度、样式以及颜色 3 方面信息。

通过 border-width 属性为边框指定宽度。其一般格式为：

```
border-width:
    border-top-width border-right-width border-bottom-width border-left-width
```

可以使用其中任何一个属性来设置相应边的宽度，而不会直接影响其他边的宽度。

通过 border-style 属性为边框指定样式。其一般格式为：

```
border-style:
    border-top-style border-right-style border-bottom-style border-left-style
```

可以使用其中任何一个属性来设置相应边的样式，而不会直接影响其他边的样式。

通过 border-color 属性为边框指定颜色。其一般格式为：

```
border-color:
    border-top-color border-right-color border-bottom-color border-left-color
```

可以使用其中任何一个属性来设置相应边的颜色，而不会直接影响其他边的颜色。

3. 设置方框的内边距

设置方框的内边距采用 padding 属性，其用法与 margin 相似。

在 CSS 中，width 和 height 指的是内容区域的宽度和高度。增加内边距、边框和外边距不会影响内容区域的尺寸，但是会增加元素框的总尺寸。

【练一练】 在 CH4 网站中添加一个 webform12.html 网页，说明 CSS 方框的使用方法。

其设计步骤如下：

① 打开 CH4 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH4”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform12.html，单击“添加”按钮。

② 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
     <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
     <title></title>
     <style type="text/css">
        div
        {   margin: 40px;   border: 20px groove #630;
            padding: 60px;  background-color: white;
            float: left;
        }
     </style>
  </head>
  <body>
```

```
    <div>
        <img src="Images\img1.bmp" style="width:150px" />
    </div>
  </body>
</html>
```

该网页在 IE 浏览器中的显示结果如图 4.28 所示，图中标识了方框各个属性的含义。

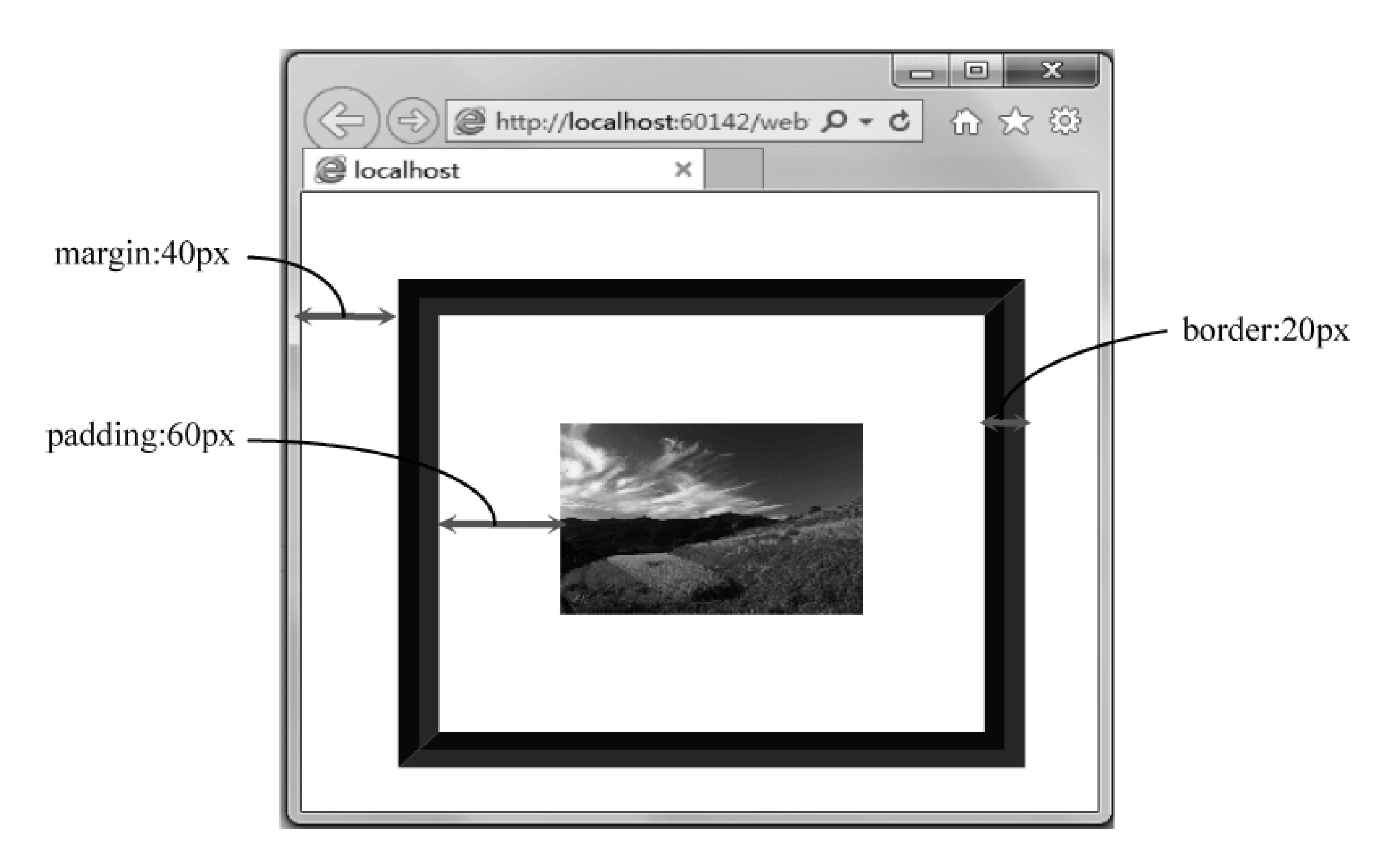


图 4.28　webform12.html 网页的显示结果

4.3.6　网页页面布局

在网页设计中，页面的整体结构布局是十分重要的，通常采用表格或方框布局。

利用表格布局主要通过将网页中的内容分为若干个区块，用表格的单元格代表区块，然后分别在不同的区块内填充内容，如图 4.29 所示是一种基本的表格布局形式。

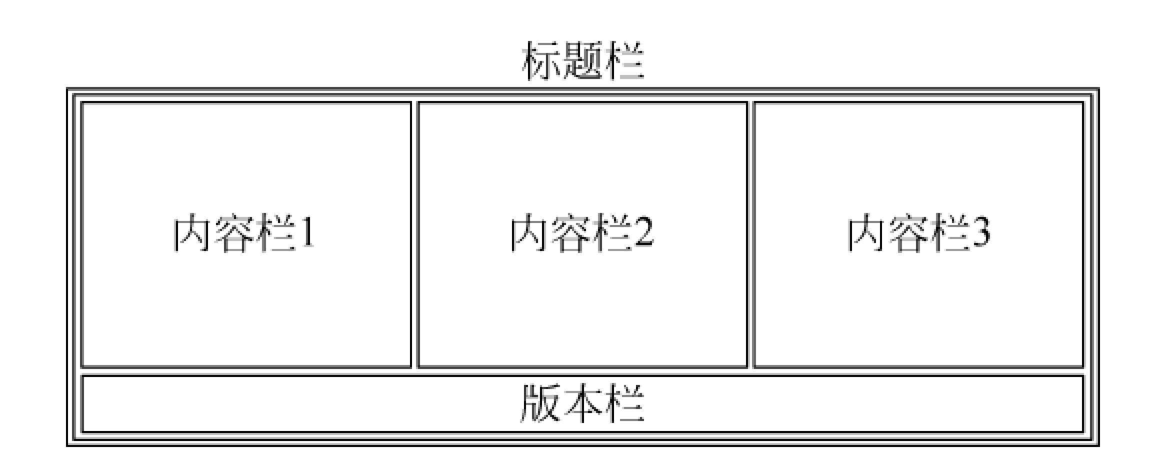


图 4.29　一种基本的表格布局形式

方框布局就是采用 DIV+CSS 进行页面布局，是 Web 2.0 时代提倡的一种页面布局方式，是一种比较灵活方便的布局方法。对于 DIV+CSS 布局的页面，浏览器会边解析边显示。DIV+CSS 网页布局的基本流程如下：

① 规划网页结构，把网站从整体上分为几个区块，规划好每个区块的大小和位置。

② 将区块用<div>标记代替，设置好每个<div>的大小和样式。

③ 通过布局属性设置<div>的位置布局。

【练一练】　在 CH4 网站中添加一个 webform13.html 网页，说明 DIV+CSS 布局方法。

其设计步骤如下：

① 打开 CH4 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH4”对话框，在中

间列表中选择“HTML 页”，将文件名称改为 webform13. html，单击“添加”按钮。

② 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <style type="text/css">
         * {margin:0px; padding:0px;}
        body
        {   font-size: 12px;    margin: 0px auto;
            height: auto;       width: 440px;
        }
        .mainBox
        {   border: 1px dashed #0099CC;
            margin: 3px;     padding: 0px;
            float: left;     height: 200px;
            width: 100px;
        }
        .mainBox h3
        {   float: left;     height: 20px;
            width: 100px;    color: #FFFFFF;
            padding: 6px 3px 3px 10px;
            background-color: #0099CC;
            font-size: 16px;
        }
        .mainBox p
        {   line-height: 1.5em;
            text-indent: 2em;
            margin: 35px 5px 5px 5px;
        }
    </style>
  </head>
  <body>
      <h2>电子商务网站开发教程</h2>
      <div class="mainBox">
          <h3>前言</h3>
          <p>正文内容</p>
      </div>
      <div class="mainBox">
          <h3>第 1 章</h3>
          <p>正文内容 </p>
      </div>
      <div class="mainBox">
          <h3>第 3 章</h3>
          <p>正文内容 </p>
      </div>
      <div class="mainBox">
          <h3>第 3 章</h3>
          <p>正文内容 </p>
      </div>
      <footer>
          <br />
          <h3>结束语</h3>
      </footer>
  </body>
</html>
```

该网页在 IE 浏览器中的显示结果如图 4.30 所示。本例采用内联样式方式设置方框选择器和类选择器等，在<body>中用<div>等标记进行页面布局。

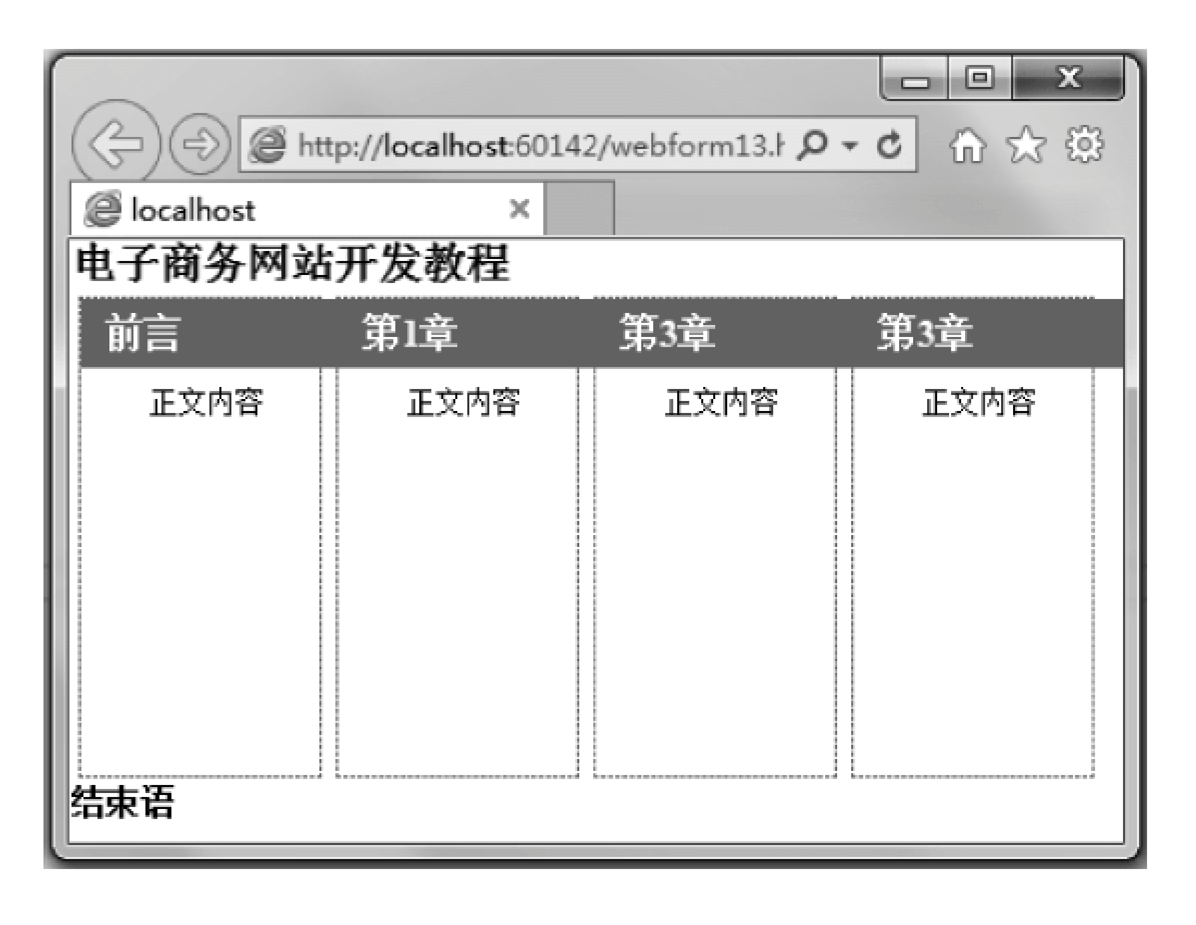


图 4.30　webform13.html 网页的执行界面

4.4 练　习　题

1. 单项选择题

(1) 下列标记不属于 HTML 文档的基本结构的是(　　)。

A. <html>　　B. <body>　　C. <head>　　D. <form>

(2) 在 HTML 中，(　　)不属于 HTML 文档的基本组成部分。

A. <style></style>　　B. <body></body>

C. <html></html>　　D. <head></head>

(3) 在 HTML 中，可以使用(　　)标记向网页中插入 GIF 动画文件。

A. <form>　　B. <body>　　C. <table>　　D. <img>

(4) 在插入图片标记中，对插入的图片进行文字说明使用的属性是(　　)。

A. name　　B. id　　C. src　　D. alt

(5) 在 HTML 上，将表单中 INPUT 元素的 TYPE 属性值设置为(　　)时，用于创建重置按钮。

A. reset　　B. set　　C. button　　D. image

(6) HTML 代码<aname="NAME">表示(　　)。

A. 创建一个超链接

B. 创建一个自动发送电子邮件的链接

C. 创建一个位于文档内部的链接点(锚点)

D. 创建一个指向位于文档内部的链接点

(7) 如果在 tmp.htm 中包含如下代码，则该 HTML 文档 IE 浏览器中打开后，用户单击此链接将(　　)。

```
<a href = "#novel">小说</a>
```

A. 使页面跳转到同一文件夹下名为 novel. html 的 HTML 文档
B. 使页面跳转到同一文件夹下名为“小说. html”的 HTML 文档
C. 使页面跳转到 tmp. htm 包含名为 novel 的锚点处
D. 使页面跳转到同一文件夹下名为“小说. html”的文档中名为 novel 的锚点处

(8) 下面关于绝对路径的说法，正确的是(　　)。
A. 绝对路径是被链接文档的完整 URL，不包括使用的传输协议
B. 使用绝对路径需要考虑源文件的位置
C. 在绝对路径中，如果目标文件被移动，则链接同样可用
D. 创建外部链接时，必须使用绝对路径

(9) 以下说法正确的是(　　)。
A. <p>标记必须以</p>标记结束
B.
标记必须以</br>标记结束
C. <title>标记应该以</title>标记结束
D. <font>标记不能在<pre>标记中使用

(10) 在 HTML 中，(　　)标记用于在网页中创建表单。
A. <input>　　B. <select>　　C. <table>　　D. <form>

(11) 对于<form action="URL" method=" * ">标记，其中 * 代表 GET 或(　　)。
A. SET　　B. PUT　　C. POST　　D. INPUT

(12) 对于标记<input id="Text1" type=" * " />，如果希望实现密码框效果，* 值是(　　)。
A. hidden　　B. text　　C. password　　D. submit

(13) 如下 HTML 代码表示(　　)。

```
<select id = "Select1" name = "D1">
    <option > A </option >
    <option > B </option >
</select >
```

A. 创建表格　　B. 创建一个滚动菜单
C. 设置每个表单项的内容　　D. 创建一个下拉菜单

(14) 执行如下 HTML 代码，当用户单击“我的链接”时，(　　)。

```
<a href = "webform1.html" target = "iframe1">我的链接</a >
<iframe name = "iframe1" ></iframe >
```

A. 在一个新窗口中显示 webform1. html 网页
B. 在 iframe1 框架中显示 webform1. html 网页
C. 在当前窗口中显示 webform1. html 网页
D. 在当前窗口的父窗口中显示 webform1. html 网页

(15) CSS 样式不包括(　　)。
A. 基于网页元素的样式　　B. 基于类的样式
C. 基于 ID 的样式　　D. 基于文件的样式

(16) 在 HTML 中，样式表按照应用方式可以分为 3 种类型，其中不包括(　　)。
A. 内嵌样式　　B. 内部样式表
C. 外部样式表文件　　D. 类样式表

(17) 在 HTML 中,以下关于 CSS 样式中文本属性的说法,错误的是(　　)。

A. font-size 用来设置文本的字体大小　B. font-family 用来设置文本的字体类型

C. color 用来设置文本的颜色　D. text-align 用来设置文本的字体形状

(18) 有关下面代码片段的说法,(　　)是正确的。

```
<style  type = "text/css">
    a {  color:blue;
       text - decoration:none; }
    a:link {  color:blue;}
    a:hover {  color:red; }
    a:visited {  color:green;}
</style>
```

A. a 样式与 a:link 样式效果相同

B. a:hover 是鼠标正在按下时链接文字的样式

C. a:link 是未被访问的链接样式

D. a:visited 是鼠标正在按下时链接文字的样式

(19) 在 HTML 中,使用 HTML 元素的 class 属性,将样式应用于网页上某个段落的代码如下所示:

```
<p class = "firstp">这是一个段落</p>
```

下面选项中,(　　)正确定义了上面代码引用的样式规则。

A. <styletype="text/css">p {color:red}</style>

B. <styletype="text/css">#firstp{color:red}</style>

C. <styletype="text/css"> p.{color:red}</style>

D. 都不正确

(20) 有关下列方框属性正确的是(　　)。

A. margin-left 是设置对象的左填充

B. border-width 是设置边框的宽度

C. padding-left 是设置内容与右边框之间的距离

D. 以上说法都不对

2. 问答题

(1) 简述 HTML 文档结构。

(2) 简述 HTML 常用的格式标记及其作用。

(3) 简述 HTML 常用的列表标记及其作用。

(4) 简述 HTML 常用的样式/节标记及其作用。

(5) 简述 HTML 常用的超链接标记及其作用。

(6) 简述<img>图像标记及其作用。

(7) 简述 iframe 框架标记及其作用。

(8) 简述在网页中创建表格的过程。

(9) 简述在网页中建立表单的过程。

(10) 简述 CSS 的作用。

(11) 简述 CSS 方框模型。

4.5　上机实验题

在 CH4 网站中设计一个名称为 Exp. html 的网页，用于输入学生信息。网页中有一个 form1 的表单，其中有一个 7×2 的表格，第 1 行两列合并，放置一个 HTML 文字"我的信息"；第 2 行两列分别为一个 HTML 文字"学号"和 HTML 文本框 Text1；第 3 行的两列分别为一个 HTML 文字"姓名"和 HTML 文本框 Text2；第 4 行的两列分别为一个 HTML 文字"性别"和两个单选按钮(Radio1 和 Rando2)；第 5 行的两列分别为一个 HTML 文字"民族"和一个下拉列表 Select1；第 6 行两列合并，放置一个＜input type＝"button"＞命令按钮 Button1 和一个＜input type＝"reset"＞命令按钮 Reset1；第 7 行两列合并，放置一个＜textarea id＝"TextArea1"＞文本域。其设计界面如图 4.31 所示。其中网页元素外观采用内嵌样式和内部样式表。内部样式表如下：

```
<style type = "text/css">
    .auto - tabstyle {                      /＊表格样式＊/
        width: 250px;
    }
    .auto - captionstyle {                  /＊"我的信息"文字的样式＊/
        font - family: 隶书;  font - size: 24px;
        color: #800080; font - weight: bold;
        text - align:center;
    }
    .auto - tagstyle {                      /＊"学号"等文字的样式＊/
        color: #0000FF; font - size: medium;
        font - weight: 700; font - family: 楷体;
        text - align:right;
    }
    #Button1, #Reset1 {                     /＊命令按钮样式＊/
       color: #FF0000; font - size: medium;
       font - weight: 700; font - family: 黑体;
    }
</style>
```

本网页的执行界面如图 4.32 所示，其中"提交"按钮上不设计任何事件处理方法，这在下一章上机实验题中添加。



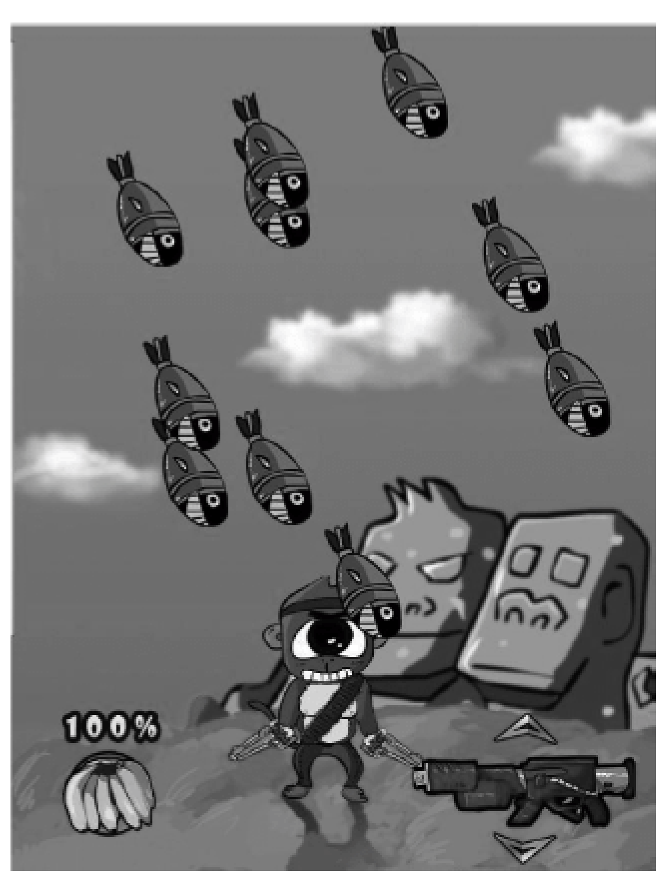
图 4.31　上机实验题网页的设计界面



图 4.32　上机实验题网页的运行界面

第 5 章　JavaScript 编程基础



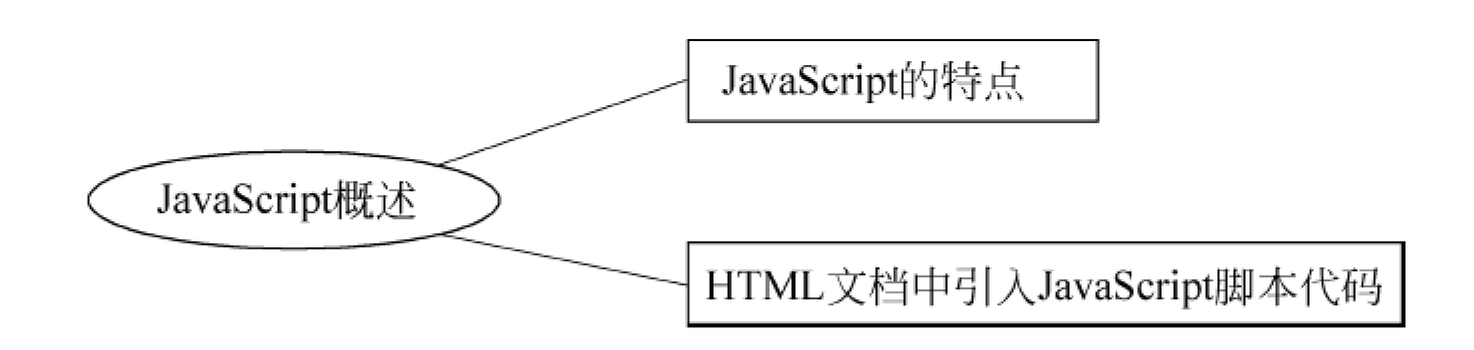


本章指南

- JavaScript概述
- JavaScript的数据类型和运算符
- JavaScript的控制语句
- JavaScript的函数设计
- 事件编程
- 浏览器对象

5.1　JavaScript 概述

知识梳理



5.1.1　JavaScript 的特点

JavaScript 是世界上最流行的脚本语言，它属于 Web 的语言，适用于 PC、笔记本电脑、平板电脑和移动电话。JavaScript 被设计为向 HTML 页面增加交互性。许多 HTML 开发者都

不是程序员，但是 JavaScript 却拥有非常简单的语法，几乎每个人都有能力将小的 JavaScript 片段添加到网页中。JavaScript 的基本特点如下：

- JavaScript 是一种属于网络的脚本语言，已经被广泛用于 Web 应用开发，常用来为网页添加各式各样的动态功能，为用户提供更流畅美观的浏览效果。通常 JavaScript 脚本是通过嵌入在 HTML 中来实现自身的功能的。
- JavaScript 是一种解释性脚本语言(代码不进行预编译)。
- JavaScript 主要用来向 HTML 网页添加交互行为。JavaScript 脚本用来实现 HTML 语言不能完成或难以完成的功能，如通过访问和操作浏览器对象控制浏览器外观、状态和运行方式等。
- JavaScript 脚本可以直接嵌入 HTML 页面，但也可以单独存储在 js 文件中有利于结构和行为的分离。
- JavaScript 具有跨平台特性，在绝大多数浏览器的支持下，可以在多种平台下运行(如 Windows、Linux、Mac、Android 和 iOS 等)。
- JavaScript 脚本语言同其他语言一样，有它自身的基本数据类型、表达式和算术运算符及程序的基本程序框架。JavaScript 提供了四种基本的数据类型和两种特殊数据类型用来处理数据和文字。而变量提供存放信息的地方，表达式则可以完成较复杂的信息处理。

5.1.2　HTML 文档中引入 JavaScript 脚本代码

在 HTML 文档中引入 JavaScript 脚本代码的常用方法如下。

(1) JavaScript 脚本代码包含于<script>和</script>标记内，嵌入到 HTML 文档中。其基本格式为：

```
<script language="javascript" type="text/javascript" src="脚本文件名">
    //JavaScript 脚本代码
</script>
```

其中，language 属性指定封装代码的脚本语言及版本；type 属性指定插入脚本代码类型；src 属性用于将外部的脚本文件内容嵌入到当前文档中。

【练一练】 在 D 盘的电子商务目录中建立一个 CH5 的子目录，将其作为网站目录，设计一个 webform1.html 网页，其功能是说明引入 JavaScript 脚本代码的方法。

其设计步骤如下：

① 启动 Visual Studio 2012。

② 选择“文件|新建|网站”命令，出现“新建网站”对话框，选择“ASP .NET 空网站”模板，选择“Web 位置”为“文件系统”，单击“浏览”按钮，选择“D:\电子商务\CH5”目录，单击“确定”按钮，创建了一个空的网站 CH5。

③ 选择“网站|添加新项”菜单命令，出现“添加新项-CH5”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform1.html，单击“添加”按钮。

④ 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
```

```
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
      <title></title>
      <script language="javascript" type="text/javascript">
          document.write("学习 JavaScript 编程!");
      </script>
    </head>
    <body>
    </body>
</html>
```

其中，<script>和</script>标记对将 Javascript 脚本代码进行封装，同时告诉浏览器其间代码为 JavaScript。

该网页在 IE 浏览器中的执行结果如图 5.1 所示。

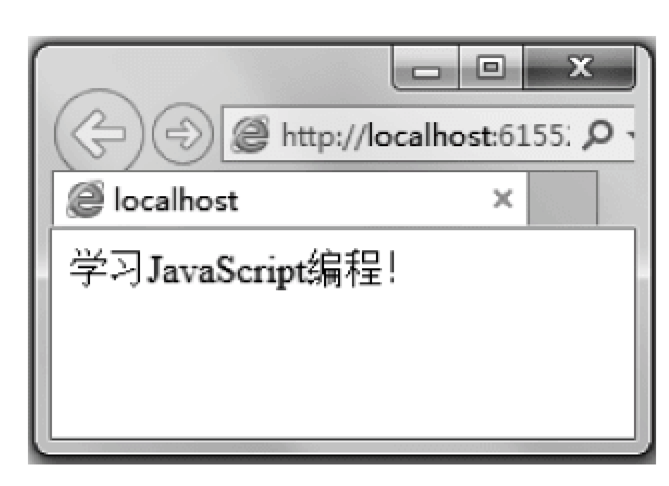


图 5.1　webform1.html 网页的执行界面

说明：JavaScript 代码用于完成所需的后台任务，最好把 JavaScript 代码放置于<head>段中，如果 JavaScript 代码用于显示内容，就把 JavaScript 代码置于<body>段中。

(2) 通过 HTML 文档事件处理方法引入 JavaScript 脚本代码。开发人员可以给 HTML 文档中设定不同的事件处理方法，通常是设置某 HTML 元素属性来引用一个脚本，属性一般以 on 开头。

【练一练】　在 CH5 网站中添加一个 webform2.html 网页，其功能是说明引入 JavaScript 脚本代码的方法。

其设计步骤如下：

① 打开 CH5 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH5”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform2.html，单击“添加”按钮。

② 进入设计视图，直接输入“输入一个字符串”文字，从工具箱的 HTML 类别拖曳两个 input(text)控件和一个 input(Button)控件，并设置它们的字体等属性。其设计界面如图 5.2 所示。



图 5.2　webform2.html 网页设计界面

③ 进入源视图，设计 JavaScript 脚本代码，完整的源视图代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <script language="javascript" type="text/javascript">
       function count()
       {     document.getElementById("Text2").value =
                "字符个数为" + document.getElementById("Text1").value.length;
       }
    </script>
```

```
    <style type = "text/css">
        .auto - style1 {
            font - family: 楷体; font - weight: bold;
            font - size: medium; color: #0000FF;  }
        #Button1 {
            color: #FF0000;   font - size: medium;
            font - weight: 700;font - family: 黑体;  }
    </style>
  </head>
  <body>
    <p><span class = "auto - style1">输入一个字符串: </span>
        <input id = "Text1" type = "text" /></p>
    <p><input id = "Button1" type = "button" value = "求字符个数" onclick = "count()"/></p>
    <p><input id = "Text2" type = "text" /></p>
  </body>
</html>
```

其中,count()方法就是一个事件处理方法,通过 Button1 控件的 onclick="count()"属性来关联它,在用户单击 Button1 时调用该事件处理方法。

④ 单击工具栏的 ▶ Internet Explorer 浏览该网页。在 Text1 文本框中输入 China,单击"求字符个数"按钮,其执行过程如图 5.3 所示。

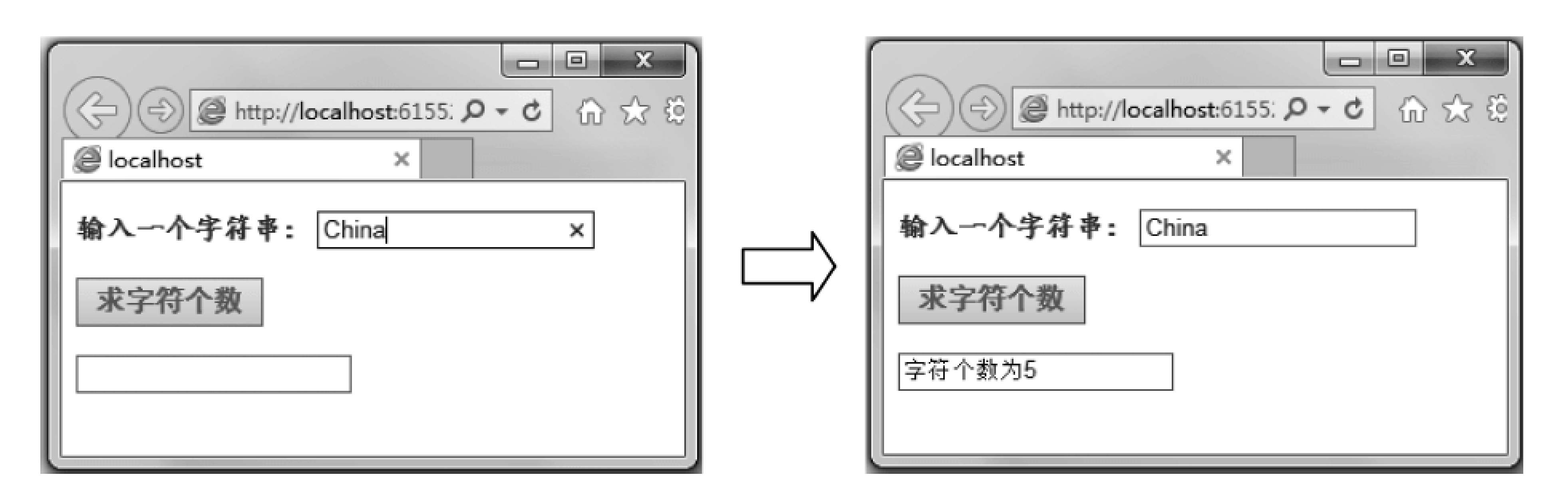


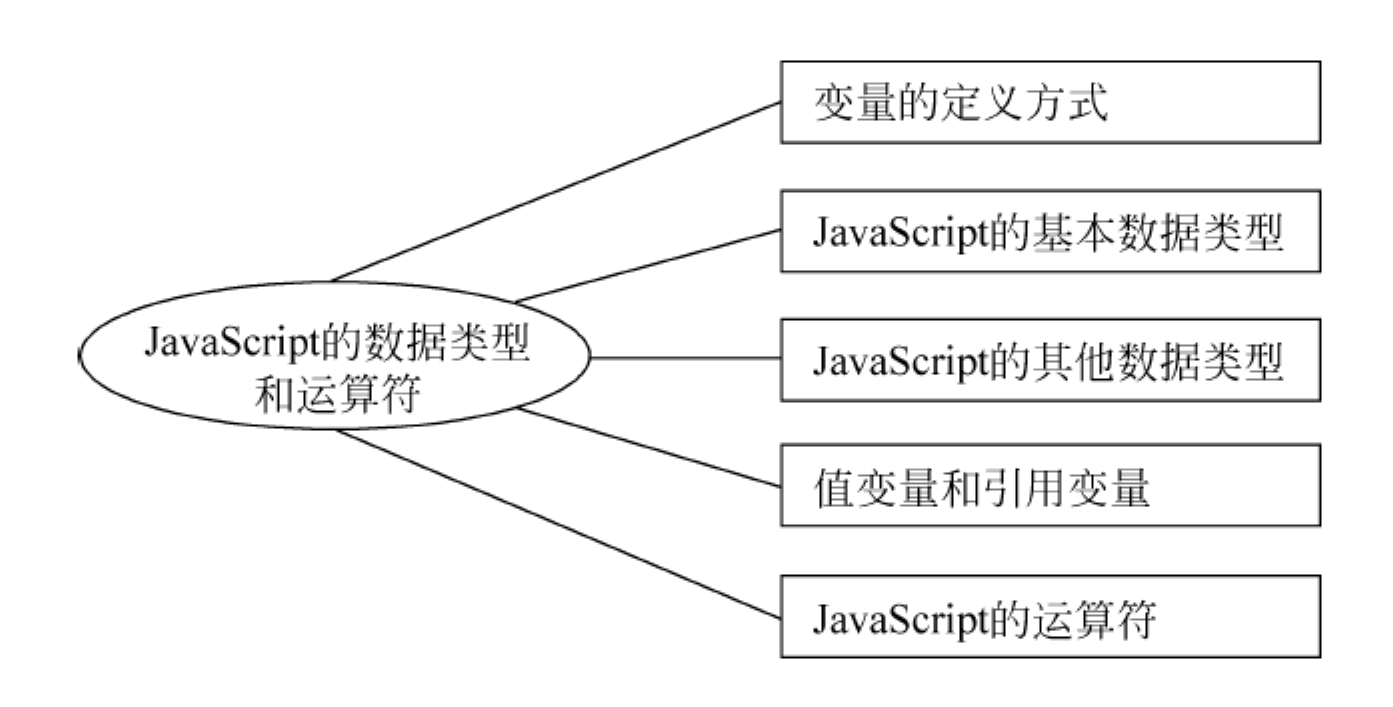
图 5.3　webform2.html 网页的执行过程

注意:在图 5.3 的两次浏览界面中右击,在出现的快捷菜单中选择"查看源"命令,显示的 HTML 文档是相同的,说明 JavaScript 脚本代码是在客户端执行的。

有关事件处理方法的内容在后面详细介绍。

5.2　JavaScript 的数据类型和运算符

知识梳理



5.2.1　变量的定义方式

变量是指程序执行中值可以发生改变的量，是程序设计语言必不可缺的部分。因为 JavaScript 程序是被浏览器顺序解释执行的，所以变量必须先定义才能使用，定义的位置必须在使用变量的语句的前面。

JavaScript 中定义变量的基本格式如下：

```
var 变量名 = 值;                    //定义变量：赋初值
变量名 = 值                         //定义变量：赋初值
```

其中，var 是 JavaScript 中用于定义变量的关键字，也可以忽略该关键字。用 var 关键字定义变量时可以不赋初值。例如：

```
var 变量名;                         //声明定义：不赋初值
```

对于没有赋初值的变量，JavaScript 认为是未定义的。只有在赋值后才能有意义。

在 JavaScript 中定义任何变量都不需要显式为其指定数据类型，系统会根据变量的值类型来确定其数据类型，所以 JavaSript 是弱类型的计算机语言。例如：

```
var n = 10;
var s = "China";
```

变量 n 的值为 10，系统会确定其数据类型为数值类型；s 的值为"China"，系统会确定其数据类型为字符串类型。

实际上，一个变量可以赋值为不同数据类型的数据。例如：

```
var n  =  10;
n  =  "China";
```

在后面使用变量 n 时，它属于字符串类型。

JavaScript 变量均为对象，当定义一个变量时，就创建了一个新的对象。

变量的命名规则是：只能以字母或"$"或"_"开头，后跟若干个字母、数字或"$"或"_"。JavaSript 变量名是区分大小写的，如 Name 和 name 是两个不同的变量。

说明：JavaScript 语句的句尾分号是非强制要求的，即句尾可以加分号，也可以不加分号，但为了清楚，通常句尾加上分号。

5.2.2　JavaScript 的基本数据类型

尽管在声明变量时不需要指定数据类型，JavaScript 的基本数据类型包括字符串型、布尔型和数值型。

1. 字符串类型

字符串是存储字符(如"China")的变量。字符串可以是引号中的任意文本。可以使用单引号或双引号。例如：

```
var mystr = "Good Bye";
var mystr = 'Good Bye';
```

2. 布尔类型

布尔(逻辑)只能有两个值：true 或 false。例如：

```
var x = true;
var y = false;
```

布尔类型常用在条件测试中。

3. 数值类型

JavaScript 只有一种数值类型，不像其他编程语言那样区分整型和浮点型。数字可以带小数点，也可以不带。

十进制整数可以采用普通记法和科学记数法。例如：

```
var n = 12;
var m = 3e7;
```

十六进制以 0x 或 0X 开头后面跟 0～F 的十六进制数字，没有指数和小数部分。例如：

```
var x = 0xAF3E;
```

八进制以 0 开头后面跟 0～7 的八进制数字，没有指数和小数部分。例如：

```
var y = 037;
```

【练一练】 在 CH5 网站中添加一个 webform3.html 网页，其功能是说明 JavaScript 的基本数据类型的应用方法。

其设计步骤如下：

① 打开 CH5 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH5”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform3.html，单击“添加”按钮。

② 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = utf-8"/>
    <title></title>
    <script language = "javascript" type = "text/javascript">
        var a = "我的数据";
        var b = false;       var c = 100;
        var d = 0x15;        var e = 015;
        var f = 0.1e-2;      var mystr = "a = " + a + "\n";
        mystr += "b = " + b + "\n";   mystr += "c = " + c + "\n";
        mystr += "d = " + d + "\n";   mystr += "e = " + e + "\n";
        mystr += "f = " + f;
        alert(mystr);
    </script>
  </head>
  <body>
  </body>
</html>
```



图 5.4 执行网页弹出的 alert 警告对话框

③ 执行该网页，弹出的 alert 警告对话框如图 5.4 所示。从中看到，<script>和</script>标记内的 JavaScript 脚本代码是按照代码顺序执行的。

说明： 调用 alert()方法会创建一个警告框，用于将浏览器或文档

的警告信息传递给客户。其参数可以是变量、字符串或表达式,警告框无返回值。alert()方法的基本语法格式为 alert("提示信息")。

5.2.3　JavaScript 的其他数据类型

JavaScript 的其他数据类型有内置对象、数组、undefined 和 null。

1. JavaScript 内置对象

在面向对象的设计模式中,将数据和处理方法捆绑在一起形成一个整体,称为对象。换句话说,对象封装了数据和操作数据的方法,要使用其中的数据或方法必须先创建该对象。可以使用 new 运算符来调用对象的构造函数,从而创建一个对象。其基本格式如下:

```
var obj = new Object();
```

其中,obj 为变量名,它指向创建的 Object 对象。要访问已经创建对象的属性或方法,可以使用“.”运算符,其基本格式如下:

```
obj.方法名(参数列表)
```

JavaScript 常用的内置对象如表 5.1 所示。

表 5.1　JavaScript 常用的内置对象

对　　象	说　　明
Object	所有对象的基对象
Array	数组对象,封装了数组的操作和属性
argument	参数对象,正在调用的函数的参数对象
Boolean	布尔对象,提供和布尔类型等价的功能
Date	日期对象,提供了日期操作和属性的对象
Error	错误对象,保存错误信息
Function	函数对象,用于创建函数
Global	全局对象,提供了全局对象函数和全局常量的对象
Math	数学对象,提供了数学函数和常量的对象
Number	数字对象,代表数值类型和提供数值常量的对象
String	字符串对象,提供了字符串操作和属性的对象

说明: NaN(或 Number.NaN)是代表非数字值的特殊值,可以把 Number 对象设置为该值,来指示其不是数字值。当方法 parseInt()和 parseFloat()不能解析指定的字符串时,就返回这个值。NaN 与其他数值进行比较的结果总是不相等的,包括它自身在内。因此,不能与 Number.NaN 比较来检测一个值是不是数字,而只能调用 isNaN()方法来比较。例如,以下代码在警告框中显示 NaN。

```
var Month = 30;
if (Month < 1 || Month > 12)
   Month = Number.NaN;
alert(Month);
```

(1) Global 对象

全局对象是预定义的对象,作为 JavaScript 的全局函数和全局属性的占位符,包括一些全局函数和全局属性。通过使用全局对象,可以访问所有其他所有预定义的对象、函数和属性。

全局对象不是任何对象的属性，所以它没有名称。

例如，如下两个类型转换方法就是 JavaScript 全局函数：

- parseInt(String)：将字符串转换为整型数值，如 parseInt("125")将字符串"125"转换为整型值 125。
- parseFloat(String)：将字符串转换为浮点型数字，如 parseInt("3.141 59")将字符串"3.141 59"转换为浮点值 3.141 59。

说明：全局对象只是一个对象，而不是类。既没有构造函数，也无法实例化一个新的全局对象。

(2) Date 对象

Date 对象用于处理日期和时间。一个 Date 对象存储的日期为自 1970 年 1 月 1 日 00:00:00 以来的毫秒数。创建 Date 对象的主要格式如下：

```
var 日期对象 = new Date();
var 日期对象 = new Date(毫秒数);
var 日期对象 = new Date (字符串);
var 日期对象 = new Date (年、月、日等参数);
```

例如：

```
var  mydate = new Date("July 29, 2007,10:30:00");
var  mydate = new Date(3580);
```

Date 对象的常用方法如表 5.2 所示。

表 5.2　Date 对象的常用方法

方　　法	说　　明
Date()	返回当日的日期和时间
getDate()	从 Date 对象返回一个月中的某一天(1～31)
getDay()	从 Date 对象返回一周中的某一天(0～6)
getMonth()	从 Date 对象返回月份(0～11)
getFullYear()	从 Date 对象以四位数字返回年份
getHours()	返回 Date 对象的小时(0～23)
getMinutes()	返回 Date 对象的分钟(0～59)
getSeconds()	返回 Date 对象的秒数(0～59)
getMilliseconds()	返回 Date 对象的毫秒(0～999)
getTime()	返回 1970 年 1 月 1 日至今的毫秒数
parse()	返回 1970 年 1 月 1 日午夜到指定日期(字符串)的毫秒数
setDate()	设置 Date 对象中月的某一天(1～31)
setMonth()	设置 Date 对象中月份(0 ～11)
setFullYear()	设置 Date 对象中的年份(4 位数字)
setHours()	设置 Date 对象中的小时(0～23)
setMinutes()	设置 Date 对象中的分钟(0～59)
setSeconds()	设置 Date 对象中的秒钟(0～59)
setMilliseconds()	设置 Date 对象中的毫秒(0～999)
setTime()	以毫秒设置 Date 对象
toString()	把 Date 对象转换为字符串
toTimeString()	把 Date 对象的时间部分转换为字符串
toDateString()	把 Date 对象的日期部分转换为字符串

【练一练】 在 CH5 网站中添加一个 webform4.html 网页，其功能是使用 Date 对象在网页上显示一个钟表。

其设计步骤如下：

① 打开 CH5 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH5”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform4.html，单击“添加”按钮。

② 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <script language="javascript">
      function disptime() {
        var time = new Date();                    //获得当前时间
        var hour = time.getHours();               //获得小时、分钟、秒
        var minute = time.getMinutes();
        var second = time.getSeconds();
        document.form1.myclock.value = hour + ":" + minute + ":" + second + " ";
        var myTime = setTimeout("disptime()", 1000);
      }
    </script>
  </head>
  <body onLoad="disptime( )">
    <form name="form1">
      <h4>当前时间: <input name="myclock" type="text"  value="" size="10"></h4>
    </form>
  </body>
</html>
```

其中，setTimeout()是属于浏览器对象 window 的方法，这里略去 window 对象名称。它用来设定时间为 1 秒，时间到了，就会执行一个指定的 disptime()方法。

③ 执行该网页，在加载网页时执行 disptime 函数，显示当前计算机时钟时间如图 5.5 所示，每隔一秒网页会自动刷新当前的时间。

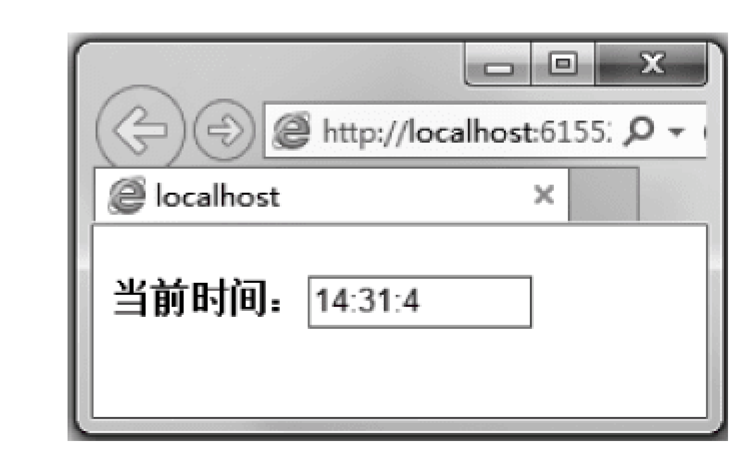


图 5.5　webform4.html 网页的执行界面

(3) String 对象

String 对象用于处理文本(字符串)。String 类定义了大量操作字符串的方法，如从字符串中提取字符或子串或检索字符或子串。

创建 String 对象的两种格式如下：

```
var s = "abc";
var mystr = String(s);
var mystr1 = new String(s);
```

其中参数 s 是要存储在 String 对象中或转换成原始字符串的值。当 String()和运算符 new 一起作为构造函数使用时，它返回一个新创建的 String 对象，存放的是字符串 s 或 s 的字符串表示。当不用 new 运算符调用 String()时，它只把 s 转换成原始的字符串，并返回转换后的值。

String 对象常用属性是 length，它返回字符串的长度。String 对象常用方法如表 5.3 所示。

表 5.3　String 对象的常用方法

方　法	说　明
concat()	连接字符串
indexOf()	检索字符串
match()	找到一个或多个正则表达式的匹配
replace()	替换与正则表达式匹配的子串
search()	检索与正则表达式相匹配的值
slice()	提取字符串的片断，并在新的字符串中返回被提取的部分
split()	把字符串分割为字符串数组
substr()	从起始索引号提取字符串中指定数目的字符
substring()	提取字符串中两个指定的索引号之间的字符
toLowerCase()	把字符串转换为小写
toUpperCase()	把字符串转换为大写

注意： JavaScript 的字符串是不可变的，String 类定义的方法都不能改变字符串的内容。像 String.toUpperCase()这样的方法，返回的是全新的字符串，而不是修改原始字符串。

(4) Math 对象

Math 对象用于执行数学任务。Math 对象能够进行比基本数学运算更为复杂的运算。Math 对象不需要生成对象的实例，可以直接访问它的属性和方法。Math 对象常用方法如表 5.4 所示。

表 5.4　Math 对象的常用方法

方　法	说　明
abs(x)	返回数的绝对值
acos(x)	返回数的反余弦值
asin(x)	返回数的反正弦值
atan(x)	以介于－PI/2 与 PI/2 弧度之间的数值来返回 x 的反正切值
atan2(y,x)	返回从 x 轴到点(x,y)的角度(介于－PI/2 与 PI/2 弧度之间)
ceil(x)	对数进行上舍入
cos(x)	返回数的余弦
exp(x)	返回 e 的指数
floor(x)	对数进行下舍入
log(x)	返回数的自然对数(底为 e)
max(x,y)	返回 x 和 y 中的最高值
min(x,y)	返回 x 和 y 中的最低值
pow(x,y)	返回 x 的 y 次幂
random()	返回 0～1 之间的随机数
round(x)	把数四舍五入为最接近的整数
sin(x)	返回数的正弦
sqrt(x)	返回数的平方根
tan(x)	返回角的正切

注意： Math 对象并不像 Date 和 String 那样是对象的类，因此没有构造函数 Math()，像 Math.sin()这样的函数只是函数，不是某个对象的方法，无须创建它，通过把 Math 作为对象使用就可以调用其所有属性和方法。

例如，使用 Math 的属性和方法的格式如下：

```
var pi_value = Math.PI;
var sqrt_value = Math.sqrt(15);
```

2. 数组

数组是对象的序列，一个数组中可以存放不同数据类型的对象。其创建与初始化的示例如下：

```
var MyArray = new Array()
var MyArray = new Array(4)
var MyArray = new Array(arg1,arg2,arg3…,argN)
var MyArray = [arg1,arg2,arg3…,argN]
```

第 1 个语句定义一个空数组，没有任何数组元素。可以用 Array 对象的 push 方法添加元素。

第 2 个语句定义一个长度为 4 的空数组，含 4 个数组元素。

第 3 个语句定义一个长度为 N、并用参数直接初始化数组元素。该方法在实际应用中最为广泛。

第 4 个语句定义一个长度为 N、并用参数直接初始化数组元素。它不调用构造函数，直接将元素放在“[]”中即可。

数组下标是基于零的，所以第一个元素是[0]，第 2 个元素是[1]，以此类推。

数组常用的属性是 length，它返回数组中元素的个数。数组常用的方法如表 5.5 所示。

表 5.5　数组的常用方法

方　法	说　明
concat()	连接两个或更多的数组，并返回结果
join()	把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔
pop()	删除并返回数组的最后一个元素
push()	向数组的末尾添加一个或更多元素，并返回新的长度
reverse()	颠倒数组中元素的顺序
shift()	删除并返回数组的第一个元素
slice()	从某个已有的数组返回选定的元素
sort()	对数组的元素进行排序
splice()	删除元素，并向数组添加新元素
toString()	把数组转换为字符串，并返回结果
unshift()	向数组的开头添加一个或更多元素，并返回新的长度

例如，执行如下函数时在 Text1 文本框中显示 5：

```
function arrlength()
{   var a = new Array(3);
    a.push(1);
    a.push("abc");
```

```
        document.getElementById("Text1").value = a.length;
    }
```

上述函数中先创建了一个含有 3 个空元素的数组 a，然后用 push 方法添加了 2 个元素(分别为整数和字符串)，所以该数组的长度为 5。

注意：不同于 C/C++和 C#中的数组，JavaScript 中的数组可以存放不同数据类型的元素。

3. undefined 和 null

undefined 这个值表示变量不含值，连空都没有，它通常用来判断一个变量是否已经定义或已经赋值。例如：

```
var s;
alert(s);
```

上述语句执行的结果如图 5.6 所示。



图 5.6　显示 undefined 的变量

每一种 JavaScript 数据类型都有自己的内容，而编程中却需要一种类型来表示“什么都没有”，null 类型就是为此目的而产生的，其表示一个空值，可以通过将变量的值设置为 null 来清空变量。例如，如下语句将 myobj 对象清空：

```
myobj = null;
```

5.2.4　值变量和引用变量

JavaScript 中，如果将一个基本数据类型的值赋给一个变量，变量中存放的是值，称为值变量；如果将一个对象赋给一个变量，变量中存放的是该对象的引用，称为引用变量。

与基本数据类型字符串、数值、布尔类型相对应的是 String 对象、Number 对象、Boolean 对象。

值变量和引用变量之间的赋值是有差异的。例如：

```
var a = 1;
var b = a;
b++;
alert(a);                                   //警告框显示 a 的值为 1
alert(b);                                   //警告框显示 a 的值为 2
```

上述程序先定义了变量 a 并赋值为 1，然后定义变量 b，赋值为 a，那么此时 b 值为 1。下一步让变量 b 自增 1。由于 a、b 都是值变量，各自存放自己的值，所以 b 值改变不会影响 a。

再看一个引用变量的例子：

```
var a = [1,2];
var b = a;
b.push(3);
alert(a);                                   //警告框显示 a 的值为 1,2,3
alert(b);                                   //警告框显示 b 的值为 1,2,3
```

上述程序先定义变量 a 并赋值数组，然后定义变量 b，并赋值为 a，那么此时 b 数组值为[1,2]。操作 b 添加新元素 3，那么 b 就为[1,2,3]，按照常理来说 a 应该不变，而结果是 a 也等于[1,2,3]。这是因为 a、b 是引用变量，执行 b=a 时让它们都指向同一数组空间。那么如何

让 a、b 各自指向自己的数组空间呢？采用的方法如下：

```
var a = [1, 2];
var b = [];
for (var i in a) {
    b.push(a[i]);
}
b.push(3);
alert(a);                                    //警告框显示 a 的值为 1,2
alert(b);                                    //警告框显示 b 的值为 1,2,3
```

5.2.5　JavaScript 的运算符

JavaScript 运算符分为算术运算符、赋值运算符、比较运算符、逻辑运算符和条件运算符几种类型。

1. JavaScript 算术运算符

JavaScript 算术运算符用于执行变量与值之间的算术运算。表 5.6（给定 a＝5）列出了 JavaScript 算术运算符及其例子。

表 5.6　JavaScript 算术运算符

运　算　符	说　　明	例　　子	结　　果
＋	加	a＝b＋2	a＝7
－	减	a＝b－2	a＝3
*	乘	a＝b * 2	a＝10
/	除	a＝b/2	a＝2.5
%	求余数(保留整数)	a＝b%2	a＝1
＋＋	累加	a＝＋＋b	a＝6
－－	递减	a＝－－b	a＝4

注意：当＋运算符用于字符串时，其功能是把文本值或字符串变量加起来(连接起来)。如果把数字与字符串相加，结果将成为字符串。

2. JavaScript 赋值运算符

JavaScript 赋值运算符用于给变量赋值。表 5.7（给定 a＝10 和 b＝5）列出了 JavaScript 赋值运算符及其例子。

表 5.7　JavaScript 赋值运算符

运　算　符	说　　明	例　　子	结　　果
＝	a＝b	a＝b	a＝5
＋＝	a＋＝b	a＝a＋b	a＝15
－＝	a－＝b	a＝a－b	a＝5
* ＝	a * ＝b	a＝a * b	a＝50
/＝	a/＝b	a＝a/b	a＝2
%＝	a%＝b	a＝a%b	a＝0

3. JavaScript 比较运算符

JavaScript 比较运算符用于测试 true 或 false。对于数值型数据，值不为 0 时表示 true，否

则为 false；对于字符型数据，值不为空时表示 true，否则为 false。

比较运算符在逻辑语句中使用，以测定变量或值是否相等。表 5.8（给定 a＝5）列出了 JavaScript 比较运算符及其例子。

表 5.8　JavaScript 比较运算符

运　算　符	说　　明	例　　子
＝＝	等于	a＝＝8 为 false
＝＝＝	全等(值和类型)	a＝＝＝5 为 true；a＝＝＝"5"为 false
!=	不等于	a!＝8 为 true
＞	大于	a＞8 为 false
＜	小于	a＜8 为 true
＞＝	大于或等于	a＞＝8 为 false
＜＝	小于或等于	a＜＝8 为 true

4. JavaScript 逻辑运算符

JavaScript 逻辑运算符也是用于测试 true 或 false。逻辑运算符用于测定变量或值之间的逻辑。表 5.9（给定 a＝6，b＝3）列出了 JavaScript 逻辑运算符及其例子。

表 5.9　JavaScript 逻辑运算符

运　算　符	说　　明	例　　子
&&	逻辑与	(a ＜10 && b ＞ 1)为 true
||	逻辑或	(a＝＝5 || b＝＝5)为 false
!	逻辑非	!(a＝＝b)为 true

5. JavaScript 条件运算符

JavaScript 还包含了基于某些条件对变量进行赋值的条件运算符。其基本格式如下：

```
变量名 = (条件表达式)?值 1:值 2;
```

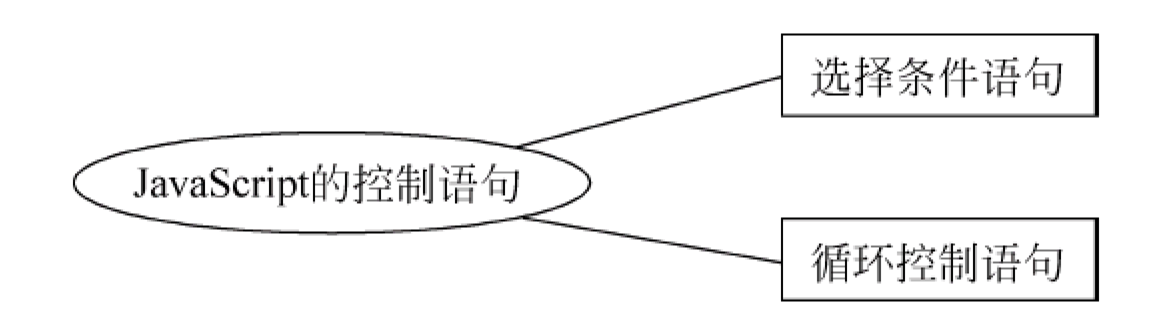
当“条件表达式”值为 true，返回值 1；否则返回值 2。例如：

```
var a = 80;
var b = (a > 60)?"及格":"不及格";
```

上述语句执行后，b 的值为“及格”。

5.3　JavaScript 的控制语句

知识梳理



5.3.1　选择条件语句

JavaScript 的选择条件语句有 if 和 switch 两种。

1. if 语句

(1) 基本 if 语句

此种格式的语法格式如下：

```
if (条件)
{   语句 1; }
[ else
{   语句 2; } ]
```

执行过程是：当条件为真时，执行语句 1；如果存在 else 部分，当条件为假时，则执行语句 2。

【练一练】 在 CH5 网站中添加一个 webform5.html 网页，其功能是求用户输入的 3 个整数中的最大值。

其设计步骤如下：

① 打开 CH5 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH5”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform5.html，单击“添加”按钮。

② 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <script language="javascript" type="text/javascript">
      function max3()
      {   var a = parseInt(document.getElementById("Text1").value);
          var b = parseInt(document.getElementById("Text2").value);
          var c = parseInt(document.getElementById("Text3").value);
          var maxv = a;
          if (a < b) maxv = b;
          if (maxv < c) maxv = c;
          document.getElementById("Text4").value = "最大值:" + maxv.toString();
      }
    </script>
    <style type="text/css">
       #Button1 {
            color: #FF0000;   font-size: medium;
            font-weight: 700; font-family: 黑体;  }
    </style>
  </head>
  <body>
    <div style="color:#0000FF;font-size:medium;font-weight:700;font-family:楷体">
        整数 1:<input id="Text1" type="text" style="width:30px" /> 
        整数 2:<input id="Text2" type="text" style="width:30px" /> 
        整数 3:<input id="Text3" type="text" style="width:30px" />
    </div>
    <div>
        <br />
        <input id="Button1" type="button" value="求最大值" onclick="max3()" /></div>
    <div>
```

```
        <br /><input id="Text4" type="text" /></div>
    </body>
</html>
```

③ 执行该网页，输入 3 个整数，单击“求最大值”命令按钮，其结果如图 5.7 所示。



图 5.7　webform5.html 网页的执行界面

（2）多重 If 语句

此种格式的语法格式如下：

```
if (条件 1)
{    语句 1; }
else if (条件 2)
{    语句 2; }
 ⋮
else if (条件 n)
{    语句 n; }
else
{    语句 n+1; }
```

执行过程是：当条件 1 为真时，执行语句 1；否则当条件 2 为真，执行语句 2；以此类推。当所有条件都为假，则执行语句 n+1。

（3）嵌套 If 语句

if 语句可以嵌套。例如：

```
if (条件 1)
{    语句 1;
     if (条件 2)
     {    语句 2;   }
}
else
{    语句 3;
     if (条件 3)
     {    语句 4;   }
}
```

2. switch 语句

像 Java、C/C++等语言中都有 switch 语句。使用 switch 语句比使用 if 语句更加方便和清晰。switch 语句的一般格式如下：

```
switch (表达式)
{
     case 值 1: 语句 1;
                 break;
     case 值 2: 语句 2;
                 break;
       ⋮
     case 值 n: 语句 n;
                 break;
     default: 语句 n+1;
}
```

每个情况(case)都是表示“如果表达式等于值 i，就执行相应的语句 i。关键字 break 会使代码跳出 switch 语句。如果没有关键字 break，代码执行就会继续进入下一个 case。关键字

default 说明了表达式的结果不等于任何一种情况时的操作(事实上,它相对于 else 子句)。

【练一练】 在 CH5 网站中添加一个 webform6. html 网页,其功能是将用户输入的分数转换成等级。

其设计步骤如下:

① 打开 CH5 网站,选择“网站|添加新项”菜单命令,出现“添加新项-CH5”对话框,在中间列表中选择“HTML 页”,将文件名称改为 webform6. html,单击“添加”按钮。

② 进入源视图,设计其 HTML 代码如下:

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <meta http - equiv = "Content - Type" content = "text/html; charset = utf - 8"/>
    <title></title>
      <script language = "javascript" type = "text/javascript">
          function trans() {
              var a = parseInt(document.getElementById("Text1").value);
              var b = Math.round(a/10 - 0.5);      //整除
              document.getElementById("Text2").value = b.toString();
              switch (b) {
                  case 9:
                  case 10: document.getElementById("Text2").value = "优"; break;
                  case 8: document.getElementById("Text2").value = "良"; break;
                  case 7: document.getElementById("Text2").value = "中"; break;
                  case 6: document.getElementById("Text2").value = "及格"; break;
                  default:document.getElementById("Text2").value = "不及格"; break;
              }
          }
     </script>
    <style type = "text/css">
        #Button1 {
            color: #FF0000;    font - size: medium;
            font - weight: 700;   font - family: 黑体;   }
    </style>
  </head>
  <body>
    <div>
       <span style = "color: #0000FF; font - size: medium; font - weight: 700;
           font - family: 楷体">分数: </span>
       <input id = "Text1" type = "text" style = "width:30px" />  
       <input id = "Button1" type = "button" value = "转换" onclick = "trans()" />
    </div>
    <div>
       <br />
       <span style = "color: #0000FF; font - size: medium; font - weight: 700;
          font - family: 楷体">等级: </span>
       <input id = "Text2" type = "text"
         style = "width:50px" />
    </div>
  </body>
</html>
```

③ 执行该网页,输入 85 分,单击“转换”按钮,其结果如图 5.8 所示。

图 5.8　webform6.html 网页的执行界面

5.3.2 循环控制语句

JavaScript 的循环控制语句有 while、do-while、for 和 for-in。

1. while 语句

while 语句的一般格式如下：

```
while (条件表达式)
{    语句;  }
```

执行过程是：先计算“条件表达式”值，若为真，执行一次语句，然后继续这一过程；否则，结束循环，执行循环后的语句。

例如，如下函数使用 while 循环在 Text1 中显示 1＋2＋…＋100 之和。

```
function comp()
{   var i = 1;
    var s = 0;
    while (i <= 100)
    {   s += i;
        i++;
    }
    document.getElementById("Text1").value = s.toString();
}
```

while 语句是前测试循环条件表达式。这意味着退出条件是在执行循环内语句之前判断的。因此，循环中的语句可能根本不被执行。

2. do-while 语句

do-while 语句的一般格式如下：

```
do
{    语句;  } while (条件表达式);
```

执行过程是：先执行一次语句，然后计算“条件表达式”值，若为真，继续这一过程；否则，结束循环，执行循环后的语句。

例如，如下函数使用 do-while 循环在 Text1 中显示 1＋2＋…＋100 之和。

```
function comp()
{   var i = 1;
    var s = 0;
    do
    {   s += i;
        i++;
    } while (i <= 100);
    document.getElementById("Text1").value = s.toString();
}
```

do-while 语句是后测试循环，即退出条件在执行循环内语句之后判断的。这意味着在计算条件表达式之前，至少会执行循环中语句一次。

3. for 语句

for 语句是前测试循环，而且在进入循环之前，能够初始化变量，并定义循环后要执行的代

码。for 语句的一般格式如下：

```
for (初始化表达式;条件表达式;后循环表达式)
{    语句;  }
```

注意：后循环表达式之后不能写分号，否则无法运行。

执行过程是：先执行初始化语句(整个循环仅执行一次)，然后计算“条件表达式”值，若为真，执行循环中的语句一次，再执行后循环表达式，继续这一过程；否则，结束循环，执行循环后的语句。

例如，如下函数使用 for 循环在 Text1 中显示 1＋2＋…＋100 之和。

```
function comp()
{   var s = 0;
    for (var i = 1; i <= 100; i++)
    {  s += i;  };
    document.getElementById("Text1").value = s.toString();
}
```

4. for-in 语句

for 语句是严格的迭代语句，用于枚举对象的属性。for-in 语句的一般格式如下：

```
for (var 迭代变量 in 枚举对象)
{
    语句;
}
```

其中，迭代变量用于枚举指定的枚举对象的属性。

【练一练】 在 CH5 网站中添加一个 webform7.html 网页，其功能是将用户输入的分数转换成等级。

其设计步骤如下：

① 打开 CH5 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH5”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform7.html，单击“添加”按钮。

② 进入设计视图，在其中拖曳一个 input(Text)的 Text1 控件。

③ 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <script language="javascript">
       function comp()
       {   var myarr = new Array();
           myarr.push(86);      myarr.push("xyz");
           myarr.push("abc");   myarr.push(125);
           var mystr = "";
           for (var i in myarr)
              mystr += i + ":" + myarr[i] + "  ";
           document.getElementById("Text1").value = mystr;
       }
    </script>
```

```
  </head>
  <body onLoad="comp()">
    <p><input id="Text1" type="text" /></p>
  </body>
</html>
```

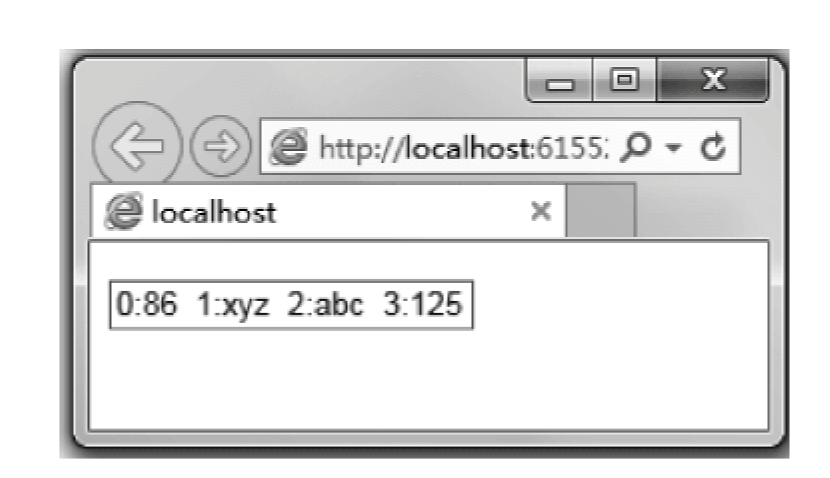


图 5.9　webform7.html 网页的执行界面

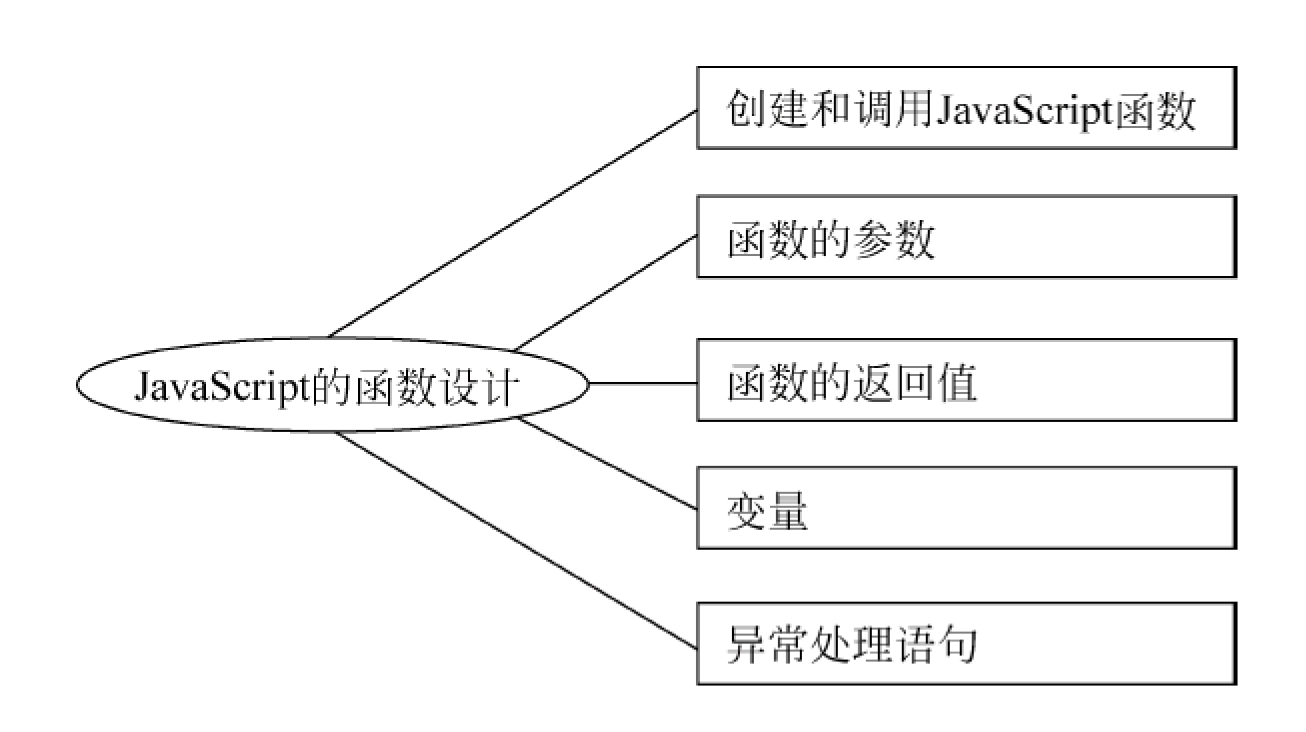
④ 执行该网页，其结果如图 5.9 所示。从中看到，for-in 语句中，迭代变量 i 为 myarr 数组的下标。

5. break 和 continue 语句

break 和 continue 语句对循环中的代码执行提供了更严格的控制。它们的不同之处是：break 语句可以立即退出循环，阻止再次反复执行任何代码。而 continue 语句只是退出当前循环，根据条件表达式还允许继续进行下一次循环。

5.4　JavaScript 的函数设计

知识梳理



5.4.1　创建和调用 JavaScript 函数

函数是由事件驱动的或者当它被调用时执行可重复使用的代码块。函数包括自定义函数和系统函数。这里主要介绍自定义函数，其创建的基本格式如下：

```
function 函数名(形参 1,形参 2,…,形参 n)
{    语句;
     return 返回值;
}
```

其中，function 是定义函数的关键字，函数的形参和 return 语句并不是必需的。函数的定义必须放在<script>元素中。

注意：函数的形参的声明不需要关键字 var，但命名与变量名命名规则相同。

在创建函数后，调用函数十分简单，其基本格式如下：

```
函数名(实参 1,实参 2,…,实参 n)
```

当调用该函数时，会执行函数内的代码。可以在某事件发生时直接调用函数(如当用户点击按钮时)，并且可由 JavaScript 在任何位置进行调用。

提示：JavaScript 对大小写敏感。关键词 function 是小写的，并且以与函数名称相同的大小写来调用函数。

5.4.2 函数的参数

函数形参也是一种变量，但这种变量只能被函数体内的语句使用，并在函数被调用时赋值。调用带形参的函数时，可以向其传递值，这些值被称为实参。函数形参和实参必须以一致的顺序出现。第一个形参就是第一个被传递的实参的给定的值，依次类推。

【练一练】 在 CH5 网站中添加一个 webform8.html 网页，其功能是用户输入学号和姓名，在一个文本框中显示用户的输入，以此说明带形参函数的使用方法。

其设计步骤如下：

① 打开 CH5 网站，选择"网站|添加新项"菜单命令，出现"添加新项-CH5"对话框，在中间列表中选择"HTML 页"，将文件名称改为 webform8.html，单击"添加"按钮。

② 进入设计视图，设计其界面如图 5.10 所示。

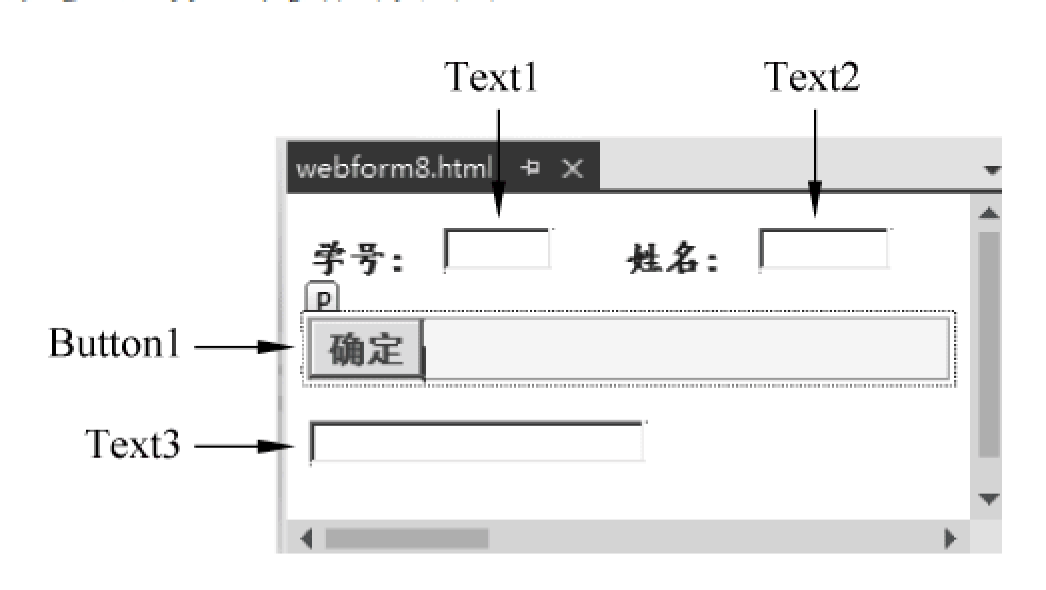


图 5.10 webform8.html 网页的设计界面

③ 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <style type="text/css">
        .auto-style1 {
            font-family: 楷体; font-weight: bold;
            font-size: medium; color: #0000FF;  }
        #Button1 {
            color: #FF0000;    font-size: medium;
            font-weight: 700; font-family: 黑体;  }
    </style>
    <script language="javascript">
        function dispstud(xh, xm) {
            var mystr = "学号: " + xh + "  姓名:" + xm;
            document.getElementById("Text3").value = mystr;
        }
    </script>
  </head>
  <body>
    <p><span class="auto-style1">学号: </span>
        <input id="Text1" type="text" style="width:40px" />
         
        <span class="auto-style1">姓名: </span>
        <input id="Text2" type="text" style="width:50px" />
    </p>
    <p><input id="Button1" type="button" value="确定"
        onclick=dispstud(Text1.value,Text2.value) /></p>
    <p><input id="Text3" type="text" /></p>
  </body>
```

```
</html>
```

④ 执行该网页,输入一个学号和一个姓名,单击“确定”按钮,其执行结果如图 5.11 所示。



图 5.11 webform8.html 网页的执行界面

在 JavaScript 中函数调用时,会自动创建一个 arguments 对象。每个函数都有一个自己的 arguments 对象,负责管理它所在函数的参数以及其他属性,包括获取的实参值。例如,设计如下函数:

```
function Sum()
{   var n = 0;
    for (var i = 0; i < arguments.length; i++)
        n += arguments[i];
    document.getElementById("Text1").value = n.toString();
}
```

调用结果如下:

```
Sum(1);                    //在文本框 Text1 中显示 1
Sum(1,2);                  //在文本框 Text1 中显示 3
Sum(1,2,3);                //在文本框 Text1 中显示 6
Sum(1,2,3,4);              //在文本框 Text1 中显示 10
```

5.4.3 函数的返回值

有时会希望函数将值返回调用它的地方,通过使用 return 语句就可以实现。在使用 return 语句时,函数会停止执行,并返回指定的值。但整个 JavaScript 并不会停止执行,仅仅是该函数,JavaScript 将从调用函数的地方继续执行代码。return 语句的基本格式如下:

```
return [表达式];
```

如果仅仅希望退出函数,可以使用不带表达式的 return 语句。

说明:如果函数无明确的返回值,或调用了没有表达式的 return 语句,那么它真正返回的值是 undefined。

函数的返回值可以是值类型,也可以是引用类型(对象)。例如,有如下函数:

```
function myFunction()
{   var x = 5;
    return x;
}
```

该函数会返回值 5。函数调用将被返回值取代,如:

```
var myVar = myFunction();
```

myVar 变量的值是 5,也就是函数 myFunction()所返回的值。

5.4.4 变量

1. 局部 JavaScript 变量

在 JavaScript 函数内部定义的变量(使用 var)是局部变量,所以只能在函数内部访问它。

可以在不同的函数中使用名称相同的局部变量，因为只有定义过该变量的函数才能识别出该变量。只要函数运行完毕，本地局部变量就会被删除。

2. 全局 JavaScript 变量

在函数外定义的变量是全局变量，网页上的所有脚本和函数都能访问它。它的存在时间也是最长的。

3. JavaScript 变量的生存期

JavaScript 变量的生命期从它们被定义的时间开始。局部变量会在函数运行以后被删除。全局变量会在页面关闭后被删除。

5.4.5 异常处理语句

JavaScript 代码在执行过程中如果出现异常，会创建一个异常类对象，该异常类对象将被提交给浏览器，这个过程称为抛出异常。当浏览器接收到该异常类对象时，会寻找能处理这一异常的代码并把当前异常对象提交给其处理。

JavaScript 中的异常可以用 try-catch-finally 语句来处理。其基本结构如下：

```
try {
    //这段代码从上往下运行，其中任何一个语句抛出异常该代码块就结束运行
}
catch (e) {
    //如果 try 代码块中抛出了异常，catch 代码块中的代码就会被执行。
    //e 是一个局部变量，用来指向 Error 对象或者其他抛出的对象
}
finally {
    //无论 try 中代码是否有异常抛出(甚至是 try 代码块中有 return 语句)，
    //finally 代码块中始终会被执行。
}
```

在 try-catch-finally 语法中，除了 try 子句以外 catch 和 finally 子句都是可选的(两者必选其一)，也就是说 try-catch-finally 语法有以下 3 种形式：

形式 1：

```
try {
  //some code
}
catch(e) {
  //somecode
}
finally {
  //some code
}
```

形式 2：

```
try {
  //some code
}
catch(e) {
  //somecode
}
```

形式 3：

```
try {
  //some code
}
finally {
  //some code
}
```

如果有 catch，一旦 try 中代码抛出异常以后就是先执行 catch 中的代码，然后执行 finally 中的代码。如果没有 catch 语句，try 中的代码抛出异常后，就会先执行 finally 中的语句，然后将 try 中抛出的异常以异常的方式继续往上抛。

不管 try 代码块的执行时如何被终止的(出现异常、return、自然终止)，finally 中的语句始终会被执行，正是由于 finally 的这种特性，通常 finally 用来执行一些清理工作。

【练一练】 在 CH5 网站中添加一个 webform9.html 网页，其功能是说明异常处理方法。

其设计步骤如下：

① 打开 CH5 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH5”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform9.html，单击“添加”按钮。

② 进入设计视图，在网页中拖曳一个 input(Button)控件 Button1，并设置其属性。

③ 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <script language="javascript">
        function message() {
           try {
                 adddlert("欢迎访问!");
           }
           catch (err) {
                var txt = "本页有一个错误。\n";
                txt += "错误描述: " + err.message + "\n";
                txt += "点击确定继续。\n";
                alert(txt);
           }
           finally {
                alert("执行 finally");
           }
        }
    </script>
  </head>
  <body>
    <p><input type="button" value="查看消息" onclick="message()"
        style="color:#FF0000;font-size:medium;font-weight:700;font-family:黑体" />
    </p>
  </body>
</html>
```

④ 执行该网页，单击“查看消息”按钮，执行 try 调用 adddlert 函数，该函数不存在，引发 catch 子句，执行其 alert 语句出现第一个警告框，单击“确定”按钮。再执行 finally 子句，执行

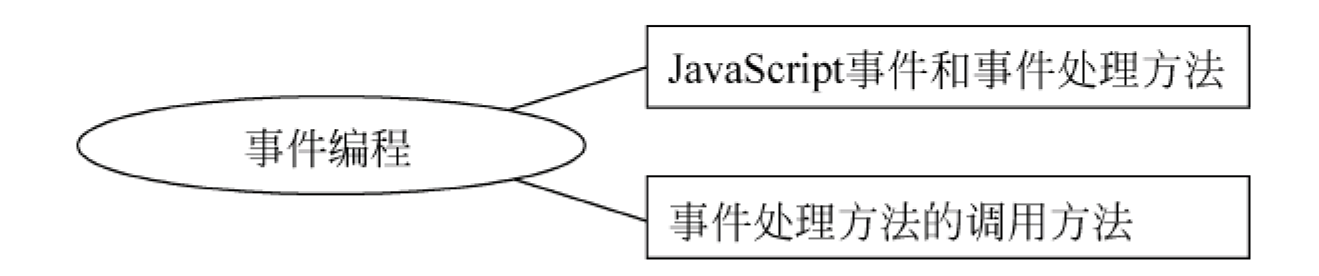
其 alert 语句出现第 2 个警告框。单击“确定”按钮返回。整个执行过程如图 5.12 所示。



图 5.12　webform9.html 网页的执行过程

5.5　事件编程

知识梳理



5.5.1　JavaScript 事件和事件处理方法

事件是能被对象识别的动作，事件可以由用户操作、程序代码或系统行为来引发。HTML 文档中的每个元素都是一个拥有属性、方法和事件的对象，称为文档对象。当引发一个文档对象的某个事件时，该对象能够按照某种方式做出响应，但具体的响应过程需要由用户编写脚本代码来实现，这种过程称为事件处理方法。

每个文档对象都拥有一个预定义的事件集。表 5.10 列出了常见的 JavaScript 事件。

表 5.10　常用的 JavaScript 事件

事件分类	事件名	说明
一般事件	onclick	鼠标点击时引发此事件
	ondblclick	鼠标双击时引发此事件
	onmousedown	按下鼠标时引发此事件
	onmouseup	鼠标按下后松开鼠标时引发此事件
	onmouseover	当鼠标移动到某对象范围的上方时引发此事件
	onmousemove	鼠标移动时引发此事件
	onmouseout	当鼠标离开某对象范围时引发此事件
	onkeypress	当键盘上的某个键被按下并且释放时引发此事件
	onkeydown	当键盘上某个按键被按下时引发此事件
	onkeyup	当键盘上某个按键被按放开时引发此事件
页面事件	onabort	图片在下载时被用户中断
	onbeforeunload	当前页面的内容将要被改变时引发此事件
	onerror	出现错误时引发此事件
	onload	网页文档加载事件

续表

事件分类	事件名	说明
页面事件	onmove	浏览器的窗口被移动时引发此事件
	onresize	当浏览器的窗口大小被改变时引发此事件
	onscroll	浏览器的滚动条位置发生变化时引发此事件
	onstop	浏览器的停止按钮被按下时引发此事件或正在下载的文件被中断
	onunload	网页文档卸载事件
表单事件	onblur	当前元素失去焦点时引发此事件
	onchange	当前元素失去焦点并且元素的内容发生改变而引发此事件
	onfocus	当某个元素获得焦点时引发此事件
	onreset	当表单中 RESET 的属性被激发时引发此事件
	onsubmit	一个表单被提交时引发此事件

5.5.2 事件处理方法的调用方法

在 JavaScript 中,可以通过多种方式来调用事件过程。下面介绍常用的方法。

1. 通过 HTML 元素的属性指定事件处理方法

在<script>标记中定义一个能够使用的 JavaScript 函数,并通过 HTML 元素的相关属性来调用该过程,这些属性为事件名称,其值为要调用的过程名称。

前面的网页示例主要采用这种方法。

2. 在标记中直接编写脚本语句

若事件过程比较简单,则在定义元素的标记中直接编写脚本语句,其中作为事件名称的属性的值包含要执行的语句,这些语句包含在单引号中,若要包含多条语句,用冒号(:)分开各个语句。

例如,如下网页就是采用这种方法。

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
  </head>
  <body>
    <p><input id="Button1" type="button" value="单击此处"
        onclick="prompt('你单击了按钮','信息提示')" /></p>
  </body>
</html>
```

其中,调用 prompt()方法显示可提示用户输入的对话框。执行该网页,单击“单击此处”按钮,出现一个对话框,如图 5.13 所示。

3. 指定特定对象的特定事件

该方法是在<script>元素中指定特定的对象以及该对象要执行的事件名称,并在<script>元素中编写事件处理代码。基本格式如下:

```
<script language="javascript" for="对象" event="事件">
```

```
    //事件处理代码
</script>
```



图 5.13 网页的执行过程

例如,有如下网页代码:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
     <script language="javascript" for="window" event="onload">
         alert("欢迎访问本网页");
    </script>
     <script language="javascript" for="Button1" event="onclick">
         alert("谢谢访问本网页");
         window.close();
    </script>
  </head>
  <body>
      <p><input id="Button1" type="button" value="关闭" /></p>
  </body>
</html>
```

执行该网页时,首先弹出"欢迎访问本网页"的对话框,用户单击"确定"按钮后,再单击网页中的"关闭"按钮,弹出"谢谢访问本网页"的对话框。

4. 通过 JavaScript 代码使用事件

该方法是在 JavaScript 脚本中直接对各对象的事件及其所调用的事件处理方法进行设置,不用在 HTML 中指定要执行的事件处理方法。设置对象事件及其事件所调用的事件处理方法的基本格式如下:

```
对象.事件=事件处理方法;
```

例如,有如下网页代码:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
```

```
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8"/>
    <title></title>
    <style type = "text/css">
        #Button1 {
            color: #FF0000;    font-size: medium;
            font-weight: 700; font-family: 黑体;
        }
    </style>
  </head>
  <body>
      <p><input id = "Button1" type = "button" value = "确定" ></p>
      <script language = "javascript" >
          function fun() {
              alert("欢迎访问本网页");
          }
          Button1.onclick = fun;
      </script>
  </body>
</html>
```

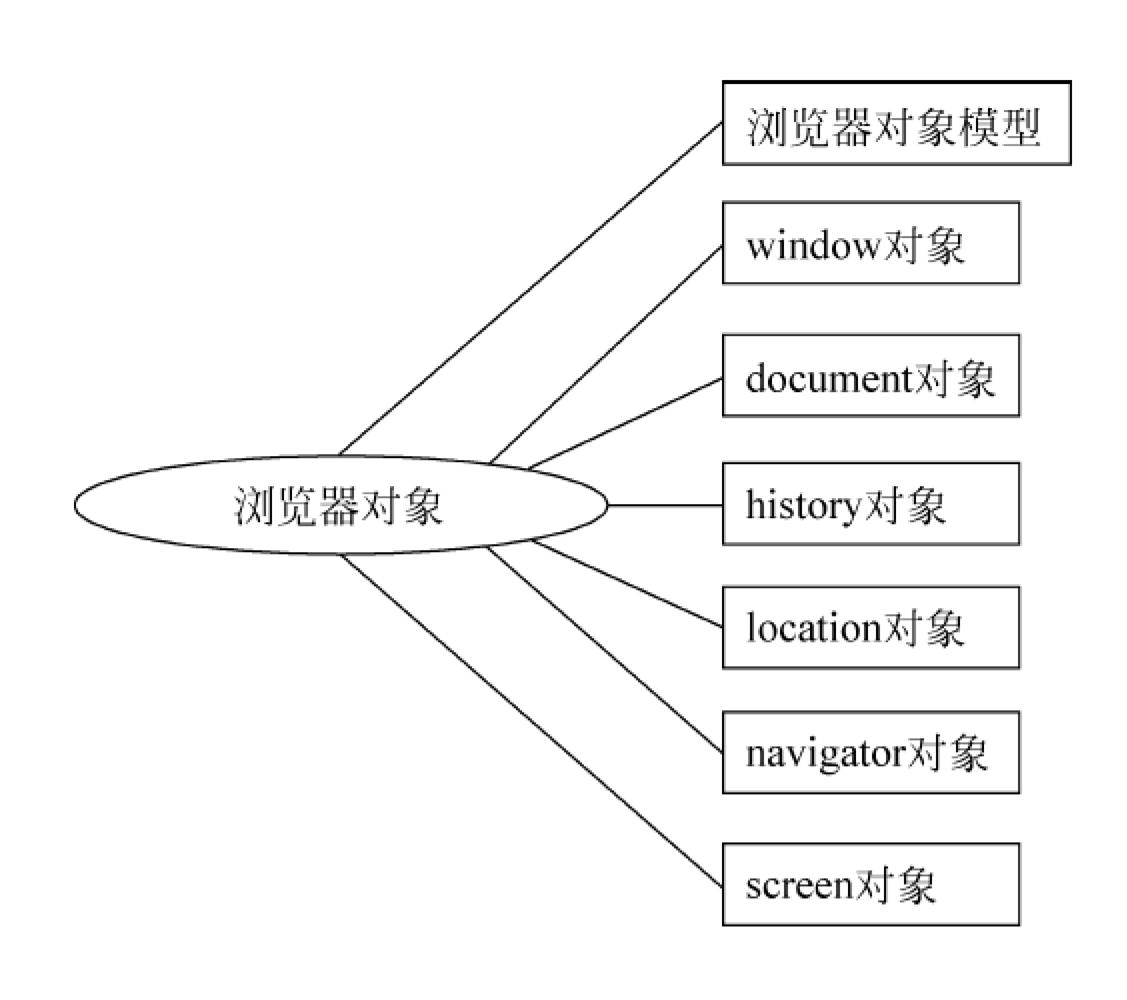
执行该网页时，用户单击“确定”按钮后弹出“欢迎访问本网页”的对话框。

上述代码中，通过 Button1.onclick ＝ fun 语句建立对象事件与事件处理方法之间的联系。需要注意的是，包含 fun 函数的 JavaScript 脚本代码应放在 Button1 元素代码的后面，不能像常规一样放在<head>元素中；否则，在执行该网页时，会出现“JavaScript 运行时错误：Button1 未定义”的错误。这是因为在执行 Button1.onclick ＝ fun 语句时网页尚未生成 Button1 对象。

另外，还可以通过<script>标记的 src 属性链接外部的 JavaScript 脚本文件和通过 JavaScript 伪 URL 地址引入 JavaScript 脚本文件，然后在 HTML 文档中调用这些脚本文件中的事件处理方法。

5.6 浏览器对象

知识梳理



5.6.1　浏览器对象模型

JavaScript 语言提供了若干个浏览器对象，在网页设计中可以通过这些对象对浏览器进行设置以及与网页控件进行交互等。这些浏览器对象构成的浏览器对象模型(Browser Object Model，BOM)。百度网页中浏览器对象的分层结构如图 5.14 所示。



图 5.14　浏览器对象的分层结构

BOM 的特点如下：

- BOM 提供了独立于内容而与浏览器窗口进行交互的对象；
- 由于 BOM 主要用于管理窗口与窗口之间的通信，因此其核心对象是 window；
- BOM 由一系列相关的对象构成，并且每个对象都提供了很多方法与属性；
- BOM 缺乏标准，JavaScript 语法的标准化组织是 ECMA，DOM 的标准化组织是 W3C；
- BOM 最初是 Netscape 浏览器标准的一部分。

5.6.2　window 对象

浏览器对象中最主要的是 window 对象，它对应着浏览器窗口本身，表示浏览器中打开的窗口。window 对象的属性和方法通常统称为 BOM。

当浏览器中同时打开多个窗口时，每个 window 对象代表一个打开的浏览器窗口，这些 window 对象之间的关系可用以下名称表示：

- window 和 self：代表当前窗口。
- top：代表主窗口，是最顶层的窗口，是所有其他窗口的父窗口。
- parent：代表当前窗口的父窗口。
- opener：由 open 方法打开的最新窗口。
- 省略 window 对象名：代表当前窗口。

window 对象常用属性如表 5.11 所示，其常用方法如表 5.12 所示，其常用事件如表 5.13 所示。

表 5.11　window 对象的常用属性

属　　性	说　　明
closed	返回窗口是否已被关闭
defaultStatus	设置或返回窗口状态栏中的默认文本
document	对 document 对象的只读引用
history	对 history 对象的只读引用
innerheight	返回窗口的文档显示区的高度
innerwidth	返回窗口的文档显示区的宽度
length	设置或返回窗口中的框架数量
location	用于窗口或框架的 location 对象
name	设置或返回窗口的名称
navigator	代表了浏览器的信息
opener	返回对创建此窗口的窗口的引用
outerheight	返回窗口的外部高度
outerwidth	返回窗口的外部宽度
pageXOffset	设置或返回当前页面相对于窗口显示区左上角的 X 位置
pageYOffset	设置或返回当前页面相对于窗口显示区左上角的 Y 位置
screen	对 screen 对象的只读引用
status	设置窗口状态栏的文本
screenLeft、screenTop、screenX、screenY	只读整数。声明了窗口的左上角在屏幕上的 X 坐标和 Y 坐标。IE、Safari 和 Opera 支持 screenLeft 和 screenTop，而 Firefox 和 Safari 支持 screenX 和 screenY

表 5.12　window 对象的常用方法

方　　法	说　　明
alert()	显示带有一段消息和一个确认按钮的警告框
blur()	把键盘焦点从顶层窗口移开
clearInterval()	取消由 setInterval()设置的 timeout
clearTimeout()	取消由 setTimeout()方法设置的 timeout
close()	关闭浏览器窗口
confirm()	显示带有一段消息以及确认按钮和取消按钮的对话框
createPopup()	创建一个弹出式窗口
focus()	把键盘焦点给予一个窗口
moveBy()	可相对窗口的当前坐标把它移动指定的像素
moveTo()	把窗口的左上角移动到一个指定的坐标
open()	打开一个新的浏览器窗口或查找一个已命名的窗口
print()	打印当前窗口的内容
prompt()	显示可提示用户输入的对话框
resizeBy()	按照指定的像素调整窗口的大小
resizeTo()	把窗口的大小调整到指定的宽度和高度
scrollBy()	按照指定的像素值来滚动内容
scrollTo()	把内容滚动到指定的坐标
setInterval()	按照指定的周期(以毫秒计)来调用函数或计算表达式
setTimeout()	在指定的毫秒数后调用函数或计算表达式

表 5.13　window 对象的常用事件

事件名	说明
onload	加载完成后引发
onunload	退出时引发
onbeforeunload	退出时引发，会发生在 onunload 之前
onhelp	显示帮助时引发
onfocus	获得焦点时引发
onblur	失去焦点时引发
onerror	错误时引发
onresize	改变大小时引发
onscroll	滚动时引发
onmove	当对象移动时引发

在客户端 JavaScript 中，window 对象是全局对象，所有的表达式都在当前的环境中计算。也就是说，要引用当前窗口根本不需要特殊的语法，可以把那个窗口的属性作为全局变量来使用。例如，可以只写 document，而不必写 window. document。

同样，可以把当前窗口对象的方法当作函数来使用，如只写 alert()，而不必写 window. alert()。

一般来说，window 对象的方法都是对浏览器窗口或框架进行某种操作。而 alert()方法、confirm()方法和 prompt 方法则不同，它们通过简单的对话框与用户进行交互。

【练一练】 在 CH5 网站中添加一个 webform10. html 网页，其功能是说明 window 对象的使用方法。

其设计步骤如下：

① 打开 CH5 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH5”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform10. html，单击“添加”按钮。

② 进入设计视图，在网页中拖曳两个 input(Button)控件，名称分别为 Button1 和 Button2，并设置其属性。

③ 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8"/>
    <title></title>
  </head>
  <body>
    <div>
        <input id = "Button1" type = "button"
            value = "放大当前窗口"
            onclick = "window.resizeBy(20, 20)"/>
        <br>
        <input id = "Button2" type = "button"
            value = "缩小当前窗口"
            onclick = "window.resizeBy( - 20, - 20)" />
    </div>
  </body>
</html>
```

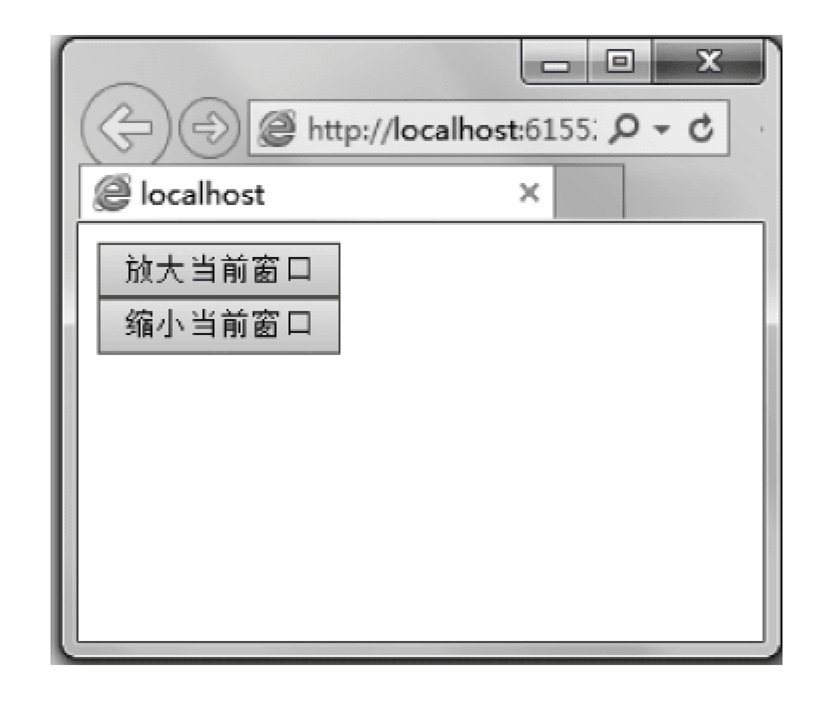


图 5.15　webform10.html 网页的执行界面

④ 执行该网页，初始界面如图 5.15 所示，单击“放大当前窗口”按钮时当前窗口以高度和宽度均为 20px

放大。单击“缩小当前窗口”按钮时当前窗口以高度和宽度均为 20px 缩小。

5.6.3 document 对象

document 对象表示浏览器窗口中显示的 HTML 文档，也就是说，每个载入浏览器的 HTML 文档都会成为 document 对象。通过该对象可以从脚本中访问到 HTML 网页中的所有元素，例如图像对象、表单对象等，从而实现与这些对象的交互。此外，利用 document 对象的属性和方法还可以控制网页的外观和内容等。

1. document 对象的属性、集合和方法

document 对象是 window 对象的一部分，可通过 window. document 属性对其进行访问。document 对象的常用属性如表 5.14 所示，其中常用的 body-主体子对象如下：

- document. body：指定文档主体的开始和结束，等价于<body></body>。
- document. body. bgColor：设置或获取对象的背景颜色。
- document. body. link：未点击过的超链接颜色。
- document. body. alink：激活超链接(焦点在此超链接时)的颜色。
- document. body. vlink：已点击过的超链接颜色。
- document. body. text：文本颜色。
- document. body. innerText：设置<body>…</body>之间的文本。
- document. body. innerHTML：设置<body>…</body>之间的 HTML 代码。
- document. body. topMargin：页面上边距。
- document. body. leftMargin：页面左边距。
- document. body. rightMargin：页面右边距。
- document. body. bottomMargin：页面下边距。
- document. body. background：背景图片。

表 5.14 document 对象的常用属性

属 性	说 明
body	提供对<body>元素的直接访问
cookie	设置或返回与当前文档有关的所有 Cookie
domain	返回当前文档的域名
lastModified	返回文档被最后修改的日期和时间
title	返回当前文档的标题
URL	返回当前文档的 URL

document 对象的常用集合如表 5.15 所示。通过集合引用的例子如下：

表 5.15 document 对象的常用集合

集 合	说 明
all[]	提供对文档中所有 HTML 元素的访问
anchors[]	返回对文档中所有 anchor 对象的引用
applets	返回对文档中所有 applet 对象的引用
forms[]	返回对文档中所有 form 对象引用
images[]	返回对文档中所有 image 对象引用
links[]	返回对文档中所有 area 和 link 对象引用

- document.images：对应页面上的 img 标记。
- document.images.length：对应页面上 img 标记的个数。
- document.images[0]：第 1 个 img 标记。
- document.images[i]：第 i−1 个 img 标记。

document 对象的常用方法如表 5.16 所示。其中 getElementById()方法可返回对拥有指定 id 的第一个对象的引用。其基本使用格式如下：

```
document.getElementById(id)
```

表 5.16 document 对象常用方法

方 法	说 明
close()	关闭用 document.open()方法打开的输出流，并显示选定的数据
getElementById()	返回对拥有指定 id 的第一个对象的引用
getElementsByName()	返回带有指定名称(name)的对象集合
getElementsByTagName()	返回带有指定标记名(type)的对象集合
open()	打开一个流，以收集来自任何 document.write()或 document.writeln()方法的输出
write()	向文档写 HTML 表达式或 JavaScript 代码
writeln()	等同于 write()方法，不同的是在每个表达式之后写一个换行符

除了 getElementById()之外，还有 getElementsByName()和 getElementsByTagName()方法。如果需要查找文档中的一个特定的元素，最有效的方法是 getElementById()。

在设计 HTML 文档的元素时，最好给元素一个 id 属性，为它指定一个(在文档中)唯一的名称，然后就可以用该 id 查找想要的元素。

如果多个语句使用“document.”前缀，可以使用 with(document)了简写。例如：

```
document.write("Good");
document.write("Bye");
```

可以简写为：

```
with(document)
{   write("Good");
    write("Bye");
}
```

2. document 对象的应用示例

示例 1：在 CH5 网站中添加一个 webform11.html 网页，其功能是说明 document 对象属性的使用方法。

其设计步骤如下：

① 打开 CH5 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH5”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform11.html，单击“添加”按钮。

② 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
```

```
    <title></title>
    <script language="javascript">
        document.write("当前打开的 HTML 文件" + "<br>");
        document.write("文件位置:" + document.URL + "<br>");
        document.write("最后修改日期:" + document.lastModified + "<br>");
    </script>
  </head>
  <body>
  </body>
</html>
```

③ 执行该网页，其结果如图 5.16 所示，显示打开文档的位置和最后修改日期的信息。

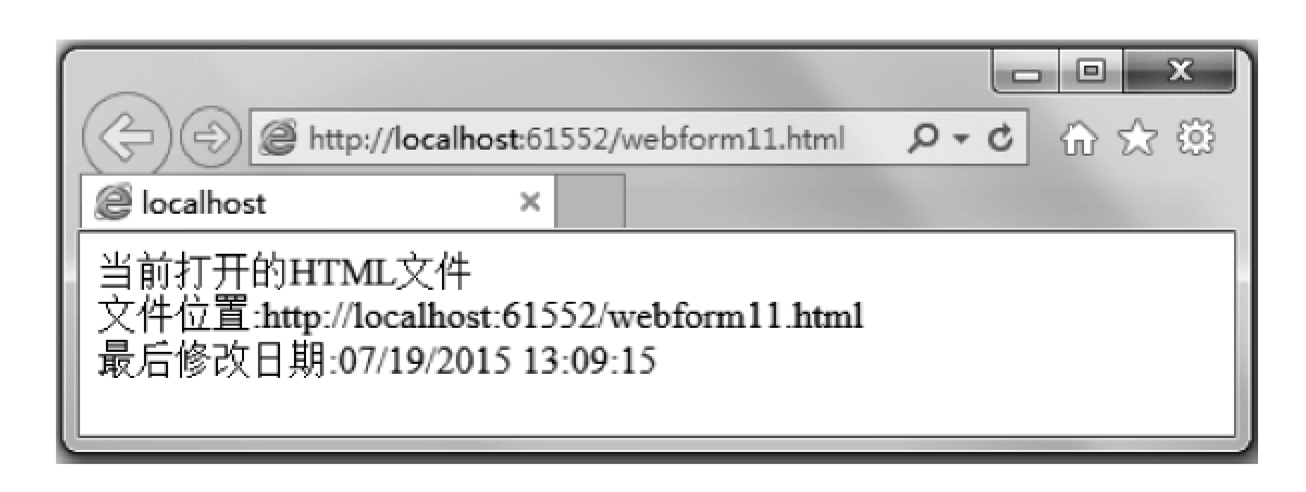


图 5.16 webform11.html 网页的执行界面

示例 2：在 CH5 网站中添加一个 Images 目录，放置若干图像文件，再添加一个 webform12.html 网页，其功能是说明 document 对象集合的使用方法。

其设计步骤如下：

① 打开 CH5 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH5”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform12.html，单击“添加”按钮。

② 进入设计视图，从工具箱的 HTML 类别向网页中拖曳一个 Select 控件、一个 input (Button)控件和一个 Image 控件，并设置其属性。

③ 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <script language="javascript">
        function dispimg() {
            var mc = document.getElementById("select1").value;
            switch (mc) {
                case "手机": document.images[0].src = "Images/1122.jpg"; break;
                case "相机": document.images[0].src = "Images/1221.jpg"; break;
                case "笔记本": document.images[0].src = "Images/2121.jpg"; break;
                case "台式机": document.images[0].src = "Images/2211.jpg"; break;
            }
        }
    </script>
    <style type="text/css">
        #Button1 {
            color: #FF0000;  font-size: medium;
            font-weight: 700;font-family: 黑体;
        }
    </style>
```

```
  </head>
  <body>
    <p><select id="select1" name="D1">
          <option>手机</option>
          <option>相机</option>
          <option>笔记本</option>
          <option>台式机</option>
       </select> 
       <input id="Button1" type="button" value="确定" onclick="dispimg()"/></p>
    <p><img alt="" /></p>
  </body>
</html>
```

其中，var mc = document.getElementById("select1").value 语句求出网页中 id 为 select1 元素的值。

④ 执行该网页，在下拉列表中选择"笔记本"，单击"确定"按钮，则在 img 元素中显示笔记本图像，如图 5.17 所示。

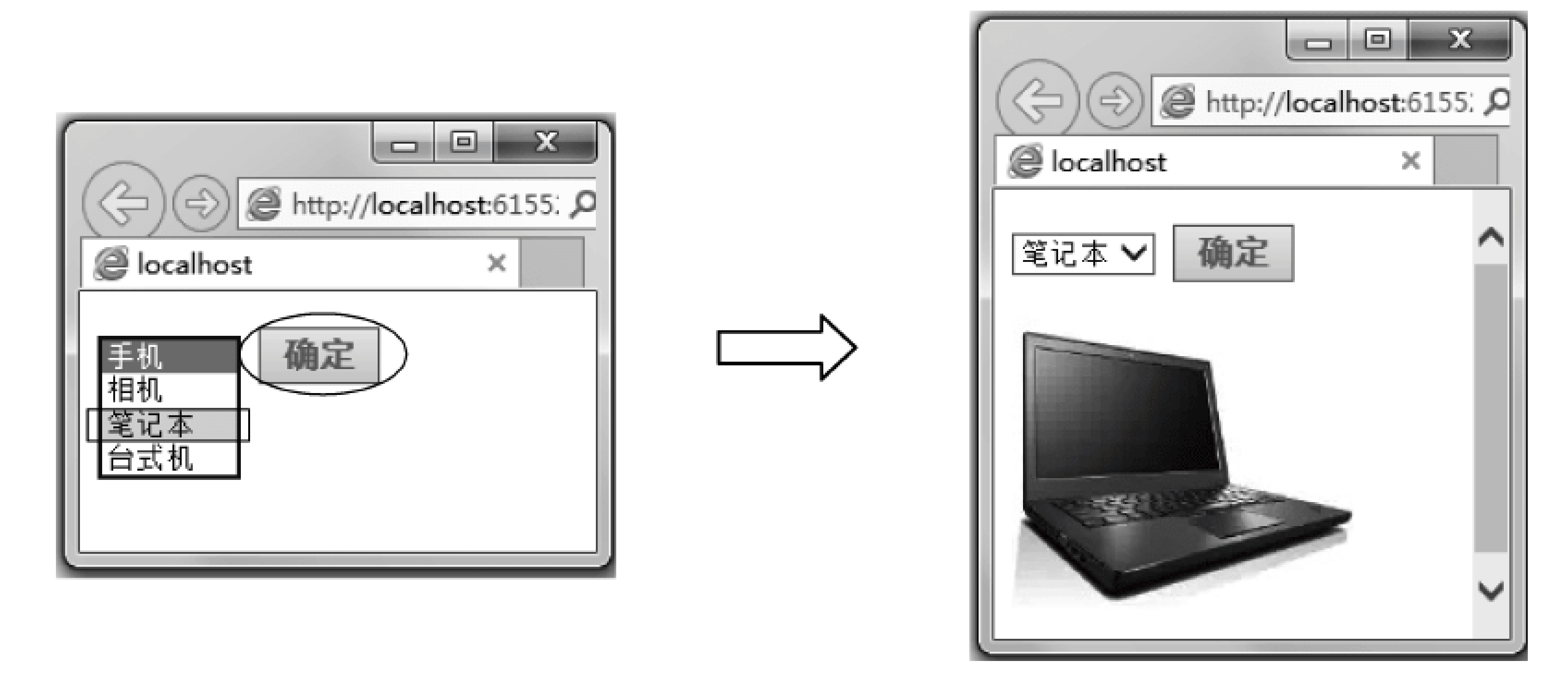


图 5.17　webform12.html 网页的执行过程

示例 3：在 CH5 网站中添加一个 webform13.html 网页，其功能是说明 document 对象集合的使用方法。

其设计步骤如下：

① 打开 CH5 网站，选择"网站|添加新项"菜单命令，出现"添加新项-CH5"对话框，在中间列表中选择"HTML 页"，将文件名称改为 webform13.html，单击"添加"按钮。

② 进入设计视图，从工具箱 HTML 类别向网页中拖曳 input(CheckBox)控件和命令按钮 Button1 和 Button2，并设置其属性。

③ 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <script language="javascript">
        function test() {
            //通过标记名查询
            var list = document.getElementsByTagName("input");
            //通过数组的 length 属性取得查询到的元素个数
            var len = list.length;
```

```
                //将每个元素的 value 属性值连接成一个字符串
                var str = "";
                for (var i = 0; i < list.length; i++) {
                    str += " " + list[i].type + ":" + list[i].value + "\n";
                }
                //输出查询到的内容
                alert("查到的元素个数: " + len + "\n\n 元素类型和值内容: \n" + str);
            }
        </script>
    </head>
    <body>
        <div id = "content">爱好:
            <input type = "checkbox" id = "ck1" value = "跑步" />跑步
            <input type = "checkbox" id = "ck2" value = "篮球" />篮球
        </div>
        <hr size = "1" color = "#FF0000" />
        <input type = "button" value = "显示网页中 input 元素" onClick = "test()"
            style = "color: #FF0000; font - size: medium; font - weight: 700;
             font - family: 黑体; width: 176px" />
    </body>
</html>
```

其中，var list＝document.getElementsByTagName("input")语句求出网页中所有 input 元素，并存储在 list 对象数组中，list[i].type 返回 list[i]对象的类型，list[i].value 返回 list[i]对象的值。

④ 执行该网页，在出现的页面中单击“显示网页中 input 元素”按钮，出现如图 5.18 所示的对话框，列出网页中所有的 input 元素和它们的值。

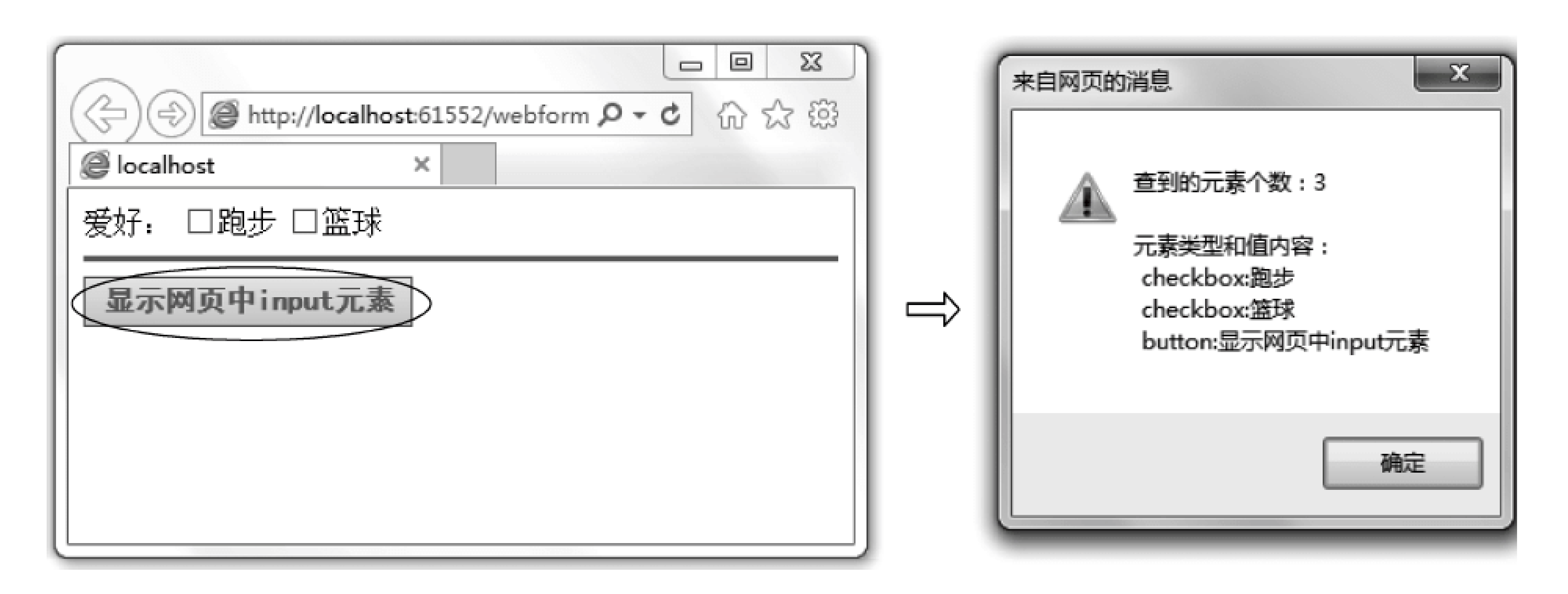


图 5.18　webform13.html 网页的执行过程

示例 4：在 CH5 网站中一个 webform14.html 网页，其功能是根据用户的输入，求一个一元二次方程的根，以此说明 document 对象引用<form>元素的使用方法。

其设计步骤如下：

① 打开 CH5 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH5”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform14.html，单击“添加”按钮。

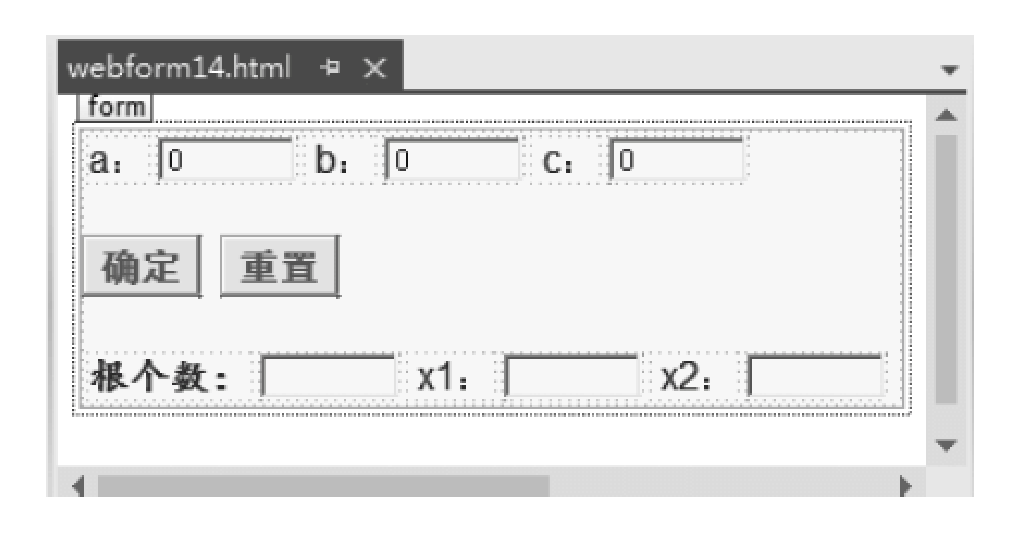


图 5.19　webform14.htm 网页的设计界面

② 进入设计视图，其设计界面如图 5.19 所示，其中有一个表单 form1，包含两个表格和两个

命令按钮,每个表格中有若干文字和文本框。

③ 进入源视图,设计其 HTML 代码如下:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <script language="javascript">
      function solve() {
        var a = parseFloat(document.form1.txta.value);          //转换为单精度数
        var b = parseFloat(document.form1.txtb.value);          //转换为单精度数
        var c = parseFloat(document.form1.txtc.value);          //转换为单精度数
        var d = b * b - 4 * a * c;
        if (d < 0)
        {   document.form1.txtn.value = 0;
            document.form1.txtx1.value = "";
            document.form1.txtx2.value = "";
        }
        else if (d == 0)
        {   document.form1.txtn.value = 1;
            var x1 = ( - b - Math.sqrt(d))/(2 * a);
            document.form1.txtx1.value = x1;
            document.form1.txtx2.value = "";
        }
        else
        {   document.form1.txtn.value = 2;
            x1 = sround(( - b - Math.sqrt(d)) / (2 * a));
            document.form1.txtx1.value = x1;
            x2 = sround(( - b + Math.sqrt(d)) / (2 * a));
            document.form1.txtx2.value = x2;
        }
      }
      function sround(x) {                                      //四舍五入并保留 2 位小数
          return Math.round(x * 100) / 100;
      }
    </script>
    <style type="text/css">
        .auto-style1 {
            font-family: Arial; font-size: medium; color: #0000FF;  }
        .auto-style2 {
            font-family: 黑体;  font-weight: bold;
            font-size: medium;  color: #FF0000;  }
        .auto-style3 {
            font-family: 楷体; font-weight: bold;
            font-size: medium; color: #0000FF; }
    </style>
  </head>
  <body>
    <form name="form1" >
      <table border="0" id="table1">
        <tr>
          <td class="auto-style1">a: </td>
          <td><input type="text" id = "txta" value="0" style="width:50px"></td>
          <td class="auto-style1"> b: </td>
```

```
            <td><input type="text" id="txtb" value="0" style="width:50px"></td>
            <td class="auto-style1"> c: </td>
            <td><input type="text" id="txtc" value="0" style="width:50px"></td>
          </tr>
        </table>
        <p><input type="button" value="确定" id="button1" onclick="solve()"
             class="auto-style2">
           <input type="reset" value="重置" id="button2" class="auto-style2"></p>
        <table border="0" id="table2">
          <tr>
            <td><span class="auto-style3">根个数: </span></td>
            <td><input type="text" id="txtn" style="width:50px"></td>
            <td class="auto-style1"> x1: </td>
            <td><input type="text" id="txtx1" style="width:50px"></td>
            <td class="auto-style1"> x2: </td>
            <td><input type="text" id="txtx2" style="width:50px"></td>
          </tr>
        </table>
      </form>
    </body>
  </html>
```

form 对象是 document 对象的一个属性，表示文档内的一个表单。由于本网页＜body＞元素中包含一个 form1 表单，所以 document. form1 引用文档中的 form1 表单，而 document. form1. txta. value 引用文档中 form1 表单中的 txta 文本框的值，也可以采用 document. forms[0]. txta. value。

④ 执行该网页，输入 a、b、c 的值，单击"确定"按钮，求出根的个数和相应的解，其结果如图 5.20 所示。



图 5.20　webform14. htm 网页的执行界面

5.6.4　history 对象

history 对象即历史对象，其中含有浏览器已访问过的网页的 URL 集合。history 对象常用属性是 length，它返回浏览器历史列表中的 URL 数量。history 对象的常用方法如表 5.17 所示。

表 5.17　history 对象的常用方法

方　法	说　明
back	载入历史列表中的前一个文档
forward	载入历史列表中的后一个文档
go	加载历史列表中的某个具体页面

例如，下面的行代码执行的操作与单击后退按钮执行的操作一样：

```
history.back();
```

又如，下面一行代码执行的操作与单击两次后退按钮执行的操作一样：

```
history.go(-2);
```

【练一练】　在 CH5 网站中一个 webform15.html 网页，其功能是说明 history 对象的使用方法。

其设计步骤如下：

① 打开 CH5 网站，选择"网站|添加新项"菜单命令，出现"添加新项-CH5"对话框，在中间列表中选择"HTML 页"，将文件名称改为 webform15.html，单击"添加"按钮。

② 进入源视图，设计其 HTML 代码如下(网页中只有一个超链接)：

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8"/>
    <title></title>
  </head>
  <body>
     <p><a href = "webform15-1.html">转向 webform15-1.html</a></p>
  </body>
</html>
```

③ 添加另一个 webform15-1.html 网页，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8"/>
    <title></title>
    <style type = "text/css">
        .auto-style1 {
            font-family: 黑体; font-weight: bold;
            font-size: medium; color: #FF0000;
        }
        .auto-style2 {
            color: #0000FF;   font-size: medium;
            font-weight: 700; font-family: 楷体;
        }
    </style>
     <script language = "javascript" >
        function fun() {
            var mystr = "学号:" + document.getElementById("Text1").value
               + "<br>姓名:" + document.getElementById("Text2").value;
            document.write(mystr);
        }
    </script>
  </head>
  <body>
    <form name = "form1" method = "post" >
       <div>
          <span class = "auto-style2">学号: </span>
            <input id = "Text1" type = "text" style = "width:60px" />
          <span class = "auto-style2">姓名: </span>
           <input id = "Text2" type = "text" style = "width:60px" />
       </div>
       <div>
           <br />
           <input id = "Button1" type = "button" value = "确定"
```

```
                class = "auto - style1" onclick = "fun()" />  
            < input id = "Button2" type = "button" value = "后退"
                class = "auto - style1" onclick = "history.back()" />  
            < input id = "Button3" type = "button" value = "前进"
                class = "auto - style1"  onclick = "history.forward()" />
        </div >
    </form >
  </body >
</html >
```

其中有 3 个命令按钮，标题分别为“确定”、“后退”和“前进”。

④ 执行 webform15. html 网页，单击“转向 webform15-1. html”超链接，出现 webform15-1. html 网页，输入学号和姓名，单击“确定”按钮，出现显示学号和姓名信息的 webform15-1. html 网页，单击 按钮返回，如图 5.21 所示。此时可以通过“后退”和“前进”按钮显示前后的网页。

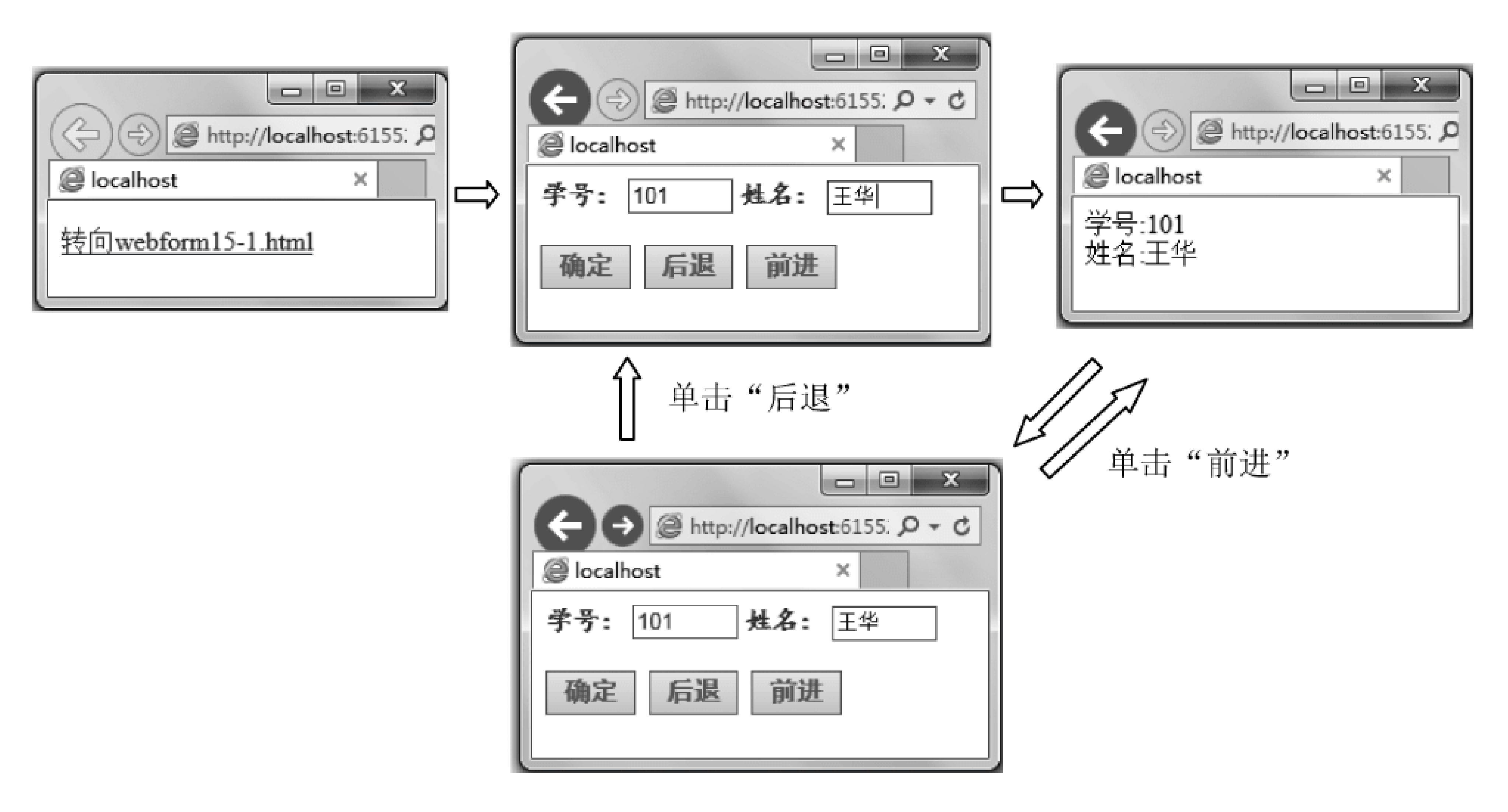


图 5.21　webform15. htm 和 webform15. htm-1 网页的执行过程

5.6.5　location 对象

location 对象存储在 window 对象的 location 属性中，表示那个窗口中当前显示的文档的 Web 地址。location 对象的常用属性如表 5.18 所示，其中 href 属性存放的是文档的完整 URL，其他属性则分别描述了 URL 的各个部分。

表 5.18　location 对象的常用属性

属　性	说　明
hash	设置或返回从井号(＃)开始的 URL(锚)
host	设置或返回主机名和当前 URL 的端口号
hostname	设置或返回当前 URL 的主机名
href	设置或返回完整的 URL
pathname	设置或返回当前 URL 的路径部分
port	设置或返回当前 URL 的端口号
protocol	设置或返回当前 URL 的协议
search	设置或返回从问号(?)开始的 URL(查询部分)

location 对象的常用方法如表 5.19 所示，其中 reload()方法可以重新加载当前文档，replace()可以加载一个新文档而无须为它创建一个新的历史记录，也就是说，在浏览器的历史列表中，新文档将替换当前文档。

表 5.19 location 对象的常用方法

方 法	说 明
assign()	加载新的文档
reload()	重新加载当前文档
replace()	用新的文档替换当前文档

【练一练】 在 CH5 网站中一个 webform16.html 网页，其功能是说明 location 对象的使用方法。

其设计步骤如下：

① 打开 CH5 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH5”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform16.html，单击“添加”按钮。

② 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
      <script language="javascript">
          window.location.assign("http://www.whu.edu.cn");
      </script>
  </head>
  <body>
  </body>
</html>
```

③ 执行本网页，当前窗口出现的武汉大学网站的主页。

5.6.6 navigator 对象

navigator 对象包含的属性描述了正在使用的浏览器，可以用 window 对象的 navigator 属性来引用它。通常使用 navigator 对象进行平台专用的配置。虽然这个对象的名称显而易见的是 Netscape 的 Navigator 浏览器，但其他实现了 JavaScript 的浏览器也支持这个对象。

navigator 对象的常用属性如表 5.20 所示。

表 5.20 navigator 对象的常用属性

属 性	说 明
appCodeName	返回浏览器的代码名
appMinorVersion	返回浏览器的次级版本
appName	返回浏览器的名称
appVersion	返回浏览器的平台和版本信息
browserLanguage	返回当前浏览器的语言
cookieEnabled	返回指明浏览器中是否启用 Cookie 的布尔值
cpuClass	返回浏览器系统的 CPU 等级
onLine	返回指明系统是否处于脱机模式的布尔值
platform	返回运行浏览器的操作系统平台
systemLanguage	返回 OS 使用的默认语言

注意：尽管没有应用于navigator对象的公开标准，不过所有浏览器都支持该对象。

【练一练】 在CH5网站中设计一个webform17.html网页，其功能是说明navigator对象的使用方法。

其设计步骤如下：

① 打开CH5网站，选择"网站|添加新项"菜单命令，出现"添加新项-CH5"对话框，在中间列表中选择"HTML页"，将文件名称改为webform17.html，单击"添加"按钮。

② 进入源视图，设计其HTML代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
      <script language="javascript">
       function fun() {
           var mystr = "";
           mystr += "浏览器名称: " + navigator.appName + "\n";
           mystr += "浏览器版本: " + parseFloat(navigator.appVersion) + "\n";
           mystr += "平台: " + navigator.platform + "\n";
           mystr += "Cookies 启用: " + navigator.cookieEnabled;
           alert(mystr);
       }
    </script>
    <style type="text/css">
        #Button1 {
            color: #FF0000;    font-size: medium;
            font-weight: 700; font-family: 黑体;  }
    </style>
  </head>
  <body>
    <p><input id="Button1" type="button" value="浏览器信息" onclick="fun()"></p>
  </body>
</html>
```

③ 执行本网页，单击"浏览器信息"按钮，显示当前浏览器的信息如图5.22所示。

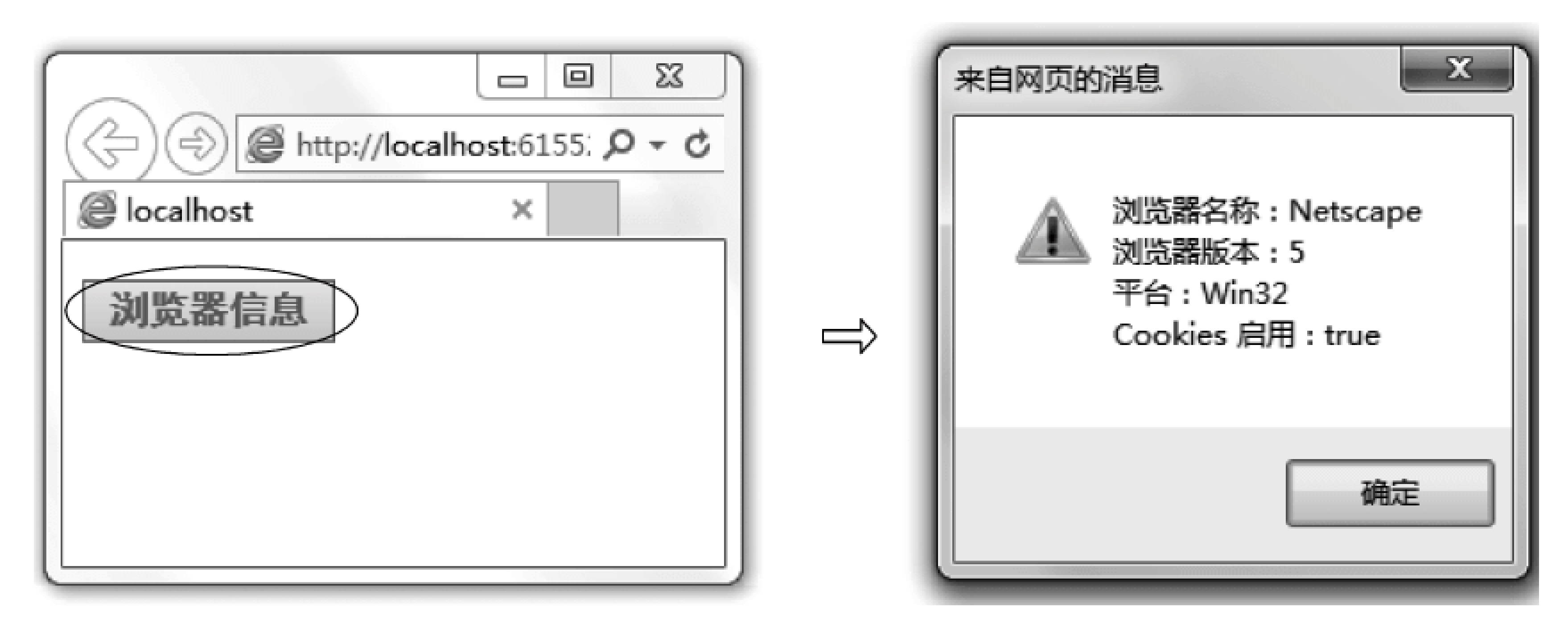


图5.22 webform17.html网页的执行过程

5.6.7 screen对象

每个window对象的screen属性都引用一个screen对象。screen对象中存放着有关显示浏览器屏幕的信息。JavaScript程序将利用这些信息来优化它们的输出，以达到用户的显示

要求。例如，一个程序可以根据显示器的尺寸选择使用大图像还是使用小图像，它还可以根据显示器的颜色深度选择使用 16 位色还是使用 8 位色的图形。另外，JavaScript 程序还能根据有关屏幕尺寸的信息将新的浏览器窗口定位在屏幕中间。

screen 对象的常用属性如表 5.21 所示。

表 5.21　screen 对象的常用属性

属　　性	说　　明
availHeight	返回显示屏幕的高度(除 Windows 任务栏之外)
availWidth	返回显示屏幕的宽度(除 Windows 任务栏之外)
bufferDepth	设置或返回调色板的位深度
colorDepth	返回目标设备或缓冲器上的调色板的深度
deviceXDPI	返回显示屏幕的每英寸水平点数
deviceYDPI	返回显示屏幕的每英寸垂直点数
height	返回显示屏幕的高度
logicalXDPI	返回显示屏幕每英寸的水平方向的常规点数
logicalYDPI	返回显示屏幕每英寸的垂直方向的常规点数
pixelDepth	返回显示屏幕的颜色分辨率
updateInterval	设置或返回屏幕的刷新率
width	返回显示器屏幕的宽度

注意：尽管没有应用于 screen 对象的公开标准，不过所有浏览器都支持该对象。

【练一练】　在 CH5 网站中一个 webform18.html 网页，其功能是说明 navigator 对象的使用方法。

其设计步骤如下：

① 打开 CH5 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH5”对话框，在中间列表中选择“HTML 页”，将文件名称改为 webform18.html，单击“添加”按钮。

② 进入源视图，设计其 HTML 代码如下：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <script language="javascript">
       document.write("屏幕信息:<br>");
       document.write("屏幕的实际高度: " + screen.availHeight + "<br>");
       document.write("屏幕的实际宽度: " + screen.availWidth + "<br>");
       document.write("屏幕区域的高度: " + screen.height + "<br>");
       document.write("屏幕区域的宽度: " + screen.width + "<br>");
    </script>
  </head>
  <body>
  </body>
</html>
```

③ 执行本网页，显示的屏幕信息如图 5.23 所示。

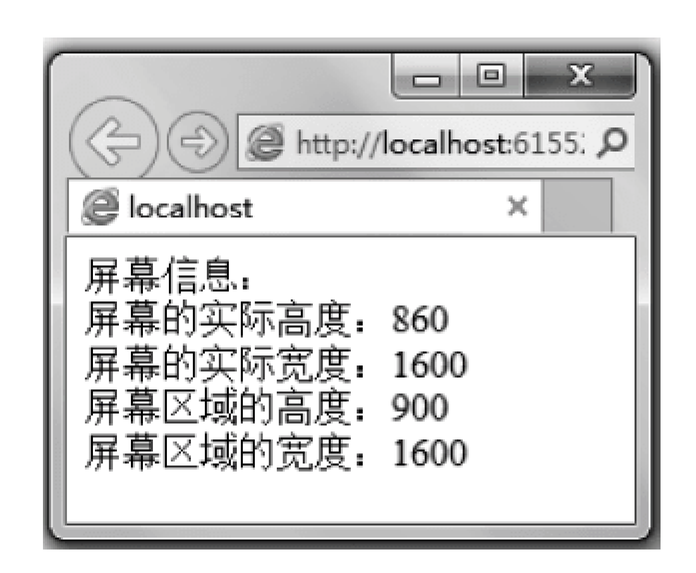


图 5.23 webform18.html 网页的执行界面

5.7 练 习 题

1. 单项选择题

(1) 在 HTML 页面中包含一个按钮控件 mybutton,如果要实现单击该按钮时调用已定义的 JavaScript 函数 compute,要编写的 HTML 代码是(　　)。

A. <inputname="mybutton" type="button" onBlur="compute()" value="计算" />

B. <inputname="mybutton" type="button" onFocus="compute()" value="计算" />

C. <inputname="mybutton" type="button" onClick="functioncompute()" value="计算" />

D. <inputname="mybutton" type="button" onClick="compute()" value="计算" />

(2) 在 HTML 页面中,定义了如下 Javascript 函数:

```
functioncompute(op){alert(op);}
```

则调用该函数时出现语法错误的 HTML 代码是(　　)。

A. <inputname="a" type="button" onclick="compute(this.value)" value="+" />

B. <inputname="b" type="button" onclick="compute('-')" value="-" />

C. <inputname="c" type="button" onclick="compute("*")" value="*" />

D. <inputname="d" type="button" onclick="compute()" value="/" />

(3) JavaScript 的基本数据类型不包括(　　)类型。

A. 字符串　　B. 数组　　C. 数值　　D. 布尔

(4) Javascript 中,(　　)语句一定会产生运行错误。

A. var_变量=NaN;　　B. var0bj=[];　　C. varobj="//";　　D. varobj={};

(5) 执行如下 JavaScript 代码,警告框的输出值是(　　)。

```
var a = 10;
b = 20;
c = 4;
alert(++b + c + a++);
```

A. 34　　B. 35　　C. 36　　D. 37

(6) 执行如下 JavaScript 代码,警告框的输出值是(　　)。

```
var x = ['abcde', 123456];
var y = typeof typeof x[1];
alert(y);
```

A. "function"　　B. "object"　　C. "number"　　D. "string"

(7) JavaScript 中,以下两个变量赋值后 alter(a==b)显示 false 的是(　　)。

A. vara=0,b=-0;　　B. vara=NaN,b=NaN;

C. vara=null,b=undefined;　　D. vara=[],b=false;

(8) JavaScript 中,以下定义变量的语句中(　　)不正确。

A. varaa;　　B. varbb=3;cc='good';

C. vardd=ee=100;　　D. varff=3,gg='he'sgood';

(9) JavaScript 中,foo 对象有 att 属性,那么以下获取 att 属性值的表达式(　　)是错误的。

A. foo.att　　B. foo["att"]

C. foo{"att"}　　D. foo["a"+"t"+"t"]

(10) (　　)作为 JavaScript 变量名是不合法。

A. string　　B. length　　C. myvar　　D. this

(11) 有如下 JavaScript 代码:

```
var arr = new Array(9);
arr[0] = 1;
arr[2] = 2;
```

则 arr 数组的 length 属性值为(　　)。

A. 2　　B. 10　　C. 8　　D. 9

(12) 有以下 JavaScript 代码,警告框的输出是(　　)。

```
var str  =  '123abc';
str  +=  str.replace('abc', '');
alert(str);
```

A. 123abc123　　B. 123abc　　C. 123　　D. abc

(13) 有以下 JavaScript 代码,警告框的输出是(　　)。

```
var a = 100,b = "100.5a6",c = "100.1";
alert(Math.max(a,b,c));
```

A. 100　　B. 100.1　　C. NaN　　D. undefined

(14) 有一个 submit 按钮,在这个按钮控件上添加(　　)事件不起作用。

A. onmouseout　　B. onmouseover　　C. onclick　　D. onsubmit

(15) (　　)语句能正确地显示结果。

A. alert(newDate(2015,12,25).getDay());

B. alert(newDate(2015,12,25,5).getDay());

C. alert(newDate(2015,12,25,5,5,9).getDay());

D. 以上三个结果都正确

(16) 以下 JavaScript 代码在警告框中显示的结果是(　　)。

```
var arr = [0, 1, 2, 3, 4, 5, 6]; arr2 = arr.slice(2, 5);
alert(arr2);
```

A. 1,2,3　　B. 1,2,3,4　　C. 2,3,4　　D. 2,3,4,5

(17) 为获取页面中多个同名对象,应使用 document 的(　　)方法。

A. getElementById()　　B. getElementsByName()

C. getElementsByTagName()　　D. 以上都不对

(18) 以下 JavaScript 代码的执行结果是(　　)。

```
<script language = "javascript">
with(document)
{   writeln("最后一次修改时间: " + document.lastModified + "<br>");
    writeln("标题: " + document.title + "<br>");
    writeln("URL: " + document.URL + "<br>");
}
</script>
```

A. 只输出最后一次修改的时间

B. 只输出文档的标题

C. 输出最后一次修改时间、文档的标题和当前的 URL

D. 什么也不输出

(19) 以下 JavaScript 代码的执行结果是(　　)。

```
<script language = "JavaScript">
    var str = "字符串";
    with(document)
    {   writeln("<b>您好,</b>");
        write("欢迎光临本网页!" + "<br>");
        writeln("<p><b>在 js 标签之间,");
        writeln(str + "可以写在这里</b></p>");
    }
</script>
```

A. 会有"
"这样的字符输出　　B. 显示两行文字,它们之间空一行

C. 最后页面会出错　　D. 会有乱码出现

(20) 以下 JavaScript 代码的说明中正确的是(　　)。

```
function fun() {
    var len = form1.text.value.length;
    alert("len = " + len);
}
```

A. 用于显示当前文档中文本框的个数

B. 用于显示当前文档中表单的个数

C. 这段代码有错误

D. 用于统计用户输入 text 文本框中字符个数

(21) 在 Javascript 中,对于浏览器对象的层次关系理解正确的是(　　)。

A. window 对象是所有页面内容的根对象

B. document 对象包含 location 对象和 history 对象

C. location 对象包含 history

D. document 对象并不包含 form 对象

(22) 以下关于浏览器对象的叙述中错误的是(　　)。

A. history 对象记录了用户在一个浏览器中已经访问过的 URL

B. location 对象相当于 IE 浏览器中的地址栏,包含关于当前 URL 地址的信息

C. location 对象是 history 对象的父对象

D. location 对象是 window 对象的子对象

2. 问答题

(1) 简述 JavaScript 和 C 中定义变量上的差异。

(2) 简述 HTML 文档中引入 JavaScript 脚本代码采用方法。

(3) 有如下网页代码,当用户单击 button 按钮时,给出其执行结果。

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>OK</title>
     <script language="javascript" type="text/javascript">
         function add(){
             var first = document.form1.first.value;
             var second = parseInt(document.form1.second.value);
             var third = parseInt(document.form1.third.value);
             alert(first + second + third);
         }
     </script>
  </head>
<body>
    <form name="form1">
        <input type="text" name="first" value="40" />
        <input type="text" name="second" value="30" />
        <input type="text" name="third" value="70" />
        <input type="button" value="add" onclick="add()" />
    </form>
  </body>
</html>
```

(4) 一个网页的<body>元素如下:

```
<body>
  <form name="form1">
    <select name="select1" size="1" onchange="fun(this)">
       <option value="a">1</option>
       <option value="b">2</option>
       <option value="c">3</option>
    </select>
  </form>
</body>
```

编写 fun 函数,当用户改变下拉列表框的选项时在一个警告框中显示当前选项的文本和值。

（5）一个网页的<body>元素如下：

```
<body>
    <form name = "form1" onsubmit = "foo();">
        <input type = "radio" name = "radioGroup" />选项 1
        <input type = "radio" name = "radioGroup" />选项 2
        <input type = "radio" name = "radioGroup" />选项 3
        <input type = "radio" name = "radioGroup" />选项 4
        <input type = "radio" name = "radioGroup" />选项 5
        <input type = "radio" name = "radioGroup" />选项 6
        <input name = "Button1" type = "submit"/>
    </form>
</body>
```

编写 foo 函数，当用户单击某个单选按钮时在一个警告框中显示当前选中的是第几个单选按钮。

（6）编写 JavaScript 函数，用户单击“求解”按钮时，在网页的文本域中显示 1～1000 之间所有能同时被 3，5，7 整除的数，并要求每行显示 6 个这样的数，如图 5.24 所示。

图 5.24　问答题第（6）题的执行界面

（7）一个 HTML 网页中有如下命令按钮和若干<input type="text">文本框：

```
<input id = "Button1" type = "button" value = "button"
onclick = "setnull()" />
```

在网页执行时单击该命令按钮会将其中所有文本框清空，编写 setnull 函数。

5.8　上机实验题

在 CH5 网站中添加一个 Exp.html 网页，其设计界面与第 4 章的上机实验题完全相同，并添加 JavaScript 代码，在用户单击“提交”按钮时在 Select1 文本域中显示输入的信息，如图 5.25 所示。



图 5.25　上机实验题网页的执行界面

第 6 章　C♯编程基础





C#语言创始人Anders Heilsberg
(安德斯·海斯博格)

Tiobe发布的全球编程语言排行榜

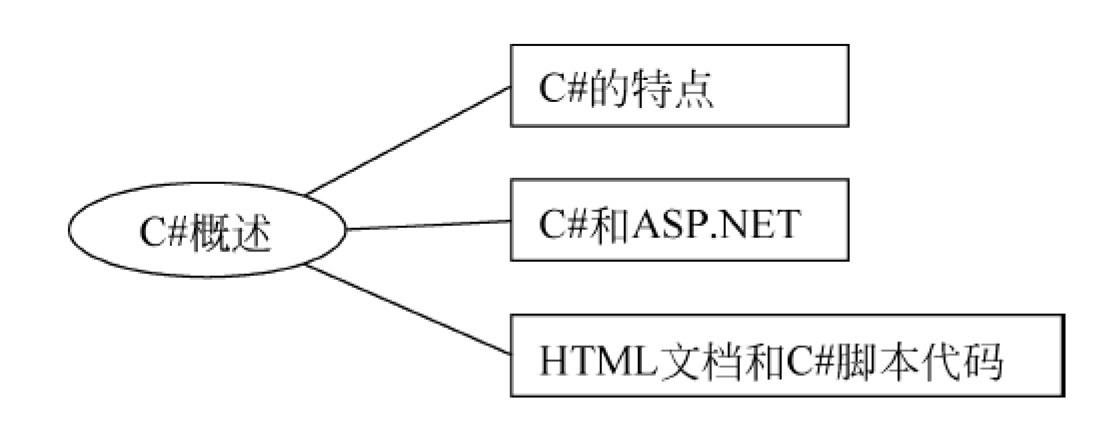
2015年5月	2014年5月	语言	使用比例
1	2	Java	16.8%
2	1	C	16.8%
3	4	C++	7.8%
4	3	Objective-C	5.4%
5	6	C#	5.3%
6	8	Python	3.7%
7	9	JavaScript	3.1%
8	11	VB.NET	2.9%
9	7	PHP	2.7%
10	-	VB	1.9%

本章指南

- C#概述
- C#的数据类型
- C#的运算符
- 结构体类型和枚举类型
- C#的控制语句
- C#的数组和集合
- 异常处理和命名空间
- 面向对象程序设计
- C#中常用类和结构体
- 继承和接口

6.1　C♯概述

知识梳理



6.1.1 C＃的特点

C＃语言是微软公司推出的面向对象的编程语言，是专门为 .NET Framework 量身打造的。程序员可以采用C＃快速方便地编写各种基于 .NET Framework 的应用程序。C＃的基本特点如下：

- 简洁的语法；
- 精心地面向对象设计；
- 与 Web 的紧密结合；
- 完整的安全性与错误处理；
- 良好的灵活性与兼容性。

6.1.2 C＃和 ASP .NET

C＃是一种编程语言。微软公司是这样描述C＃的："C＃是从C和C++派生来的一种简单、现代、面向对象和类型安全的编程语言。C＃主要是从C/C++编程语言家族移植过来的，C和C++的程序员会马上熟悉它。C＃试图结合VB的快速开发能力和C++强大灵活的能力。"

ASP .NET 不是一种语言，而是创建动态 Web 网页的一种强大的服务器端技术，它是 .NET Framework 中一套用于生成 Web 应用程序和 Web 服务的技术。ASP .NET 网页在服务器上执行，并生成发送到浏览器的标记(如 HTML、XML 或 WML)。可以使用任何 .NET 兼容语言(如 C＃或 VB)编写 Web 服务文件中的服务器端逻辑(而不是客户端逻辑，而 JavaScript 用于编写客户端逻辑)。ASP .NET 网页使用一种由事件驱动、已编译的编程模型，这种模型可以提高性能并支持将用户界面层同应用程序逻辑层相隔离。

显然，C＃作为 .NET Framework 的一种兼容语言，可以作为 ASP .NET 网页编程的脚本语言，网页运行时由 ASP .NET 引擎执行其脚本，在生成最终的 HTML 网页后由服务器发送给客户端。实际上，ASP .NET 4.5 本身就采用C＃语言开发的，所以C＃不仅适用于 Web 应用程序的开发，也适用于开发强大的系统程序。

6.1.3 HTML 文档和C＃脚本代码

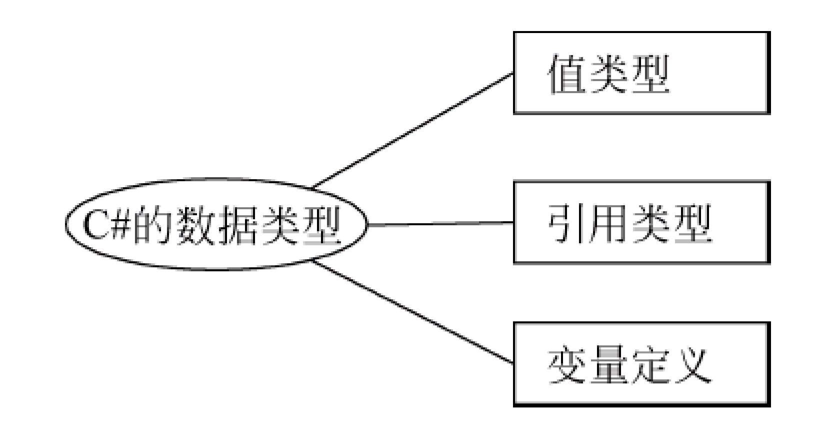
JavaScript 脚本代码属于客户端代码，Web 服务器将 JavaScript 脚本代码和 HTML 文档一起发送给客户端，由客户端浏览器来执行。

而C＃脚本代码不是在客户端执行的，是在服务器端执行的。采用 Visual Studio 设计的网页是 ASP .NET 网页，其中可以包含C＃脚本代码，由 ASP .NET 引擎执行C＃脚本代码并进行转换，最终生成 HTML 文档，再发送给客户端。所以，HTML 文档可以看成是由C＃脚本代码执行的结果。

开发人员主要是设计 ASP .NET 网页。如果采用单文件页模型，C＃脚本代码包含在<script runat="server">和</script>中；如果采用代码隐藏模型，C＃脚本代码包含的单独的.aspx.cs 文件中，通过网页中的<%@ Page … %>页面指令进行关联。

6.2　C＃的数据类型

知识梳理



6.2.1　值类型

C＃数据类型主要分为值类型和引用类型两大类。值类型的变量内含变量值本身，C＃的值类型可以分为简单类型、结构体类型和枚举类型等。

1. 整数类型

整数类型变量的值为整数。数学上的整数可以从负无穷大到正无穷大，但是由于计算机的存储单元是有限的，所以计算机语言提供的整数类型的值总是在一定的范围之内。具体各整数类型及其取值范围如表 6.1 所示。

表 6.1　整数类型及其取值范围

类型标识符	说明	占用位数	取值范围	示例
sbyte	带符号字节型	8	－128～127	sbyte i＝10；
byte	无符号字节型	8	0～255	byte i＝10；
short	带符号短整型	16	－32 768～32 767	short i＝10；
ushort	无符号短整型	16	0～65 535	ushort i＝10；
int	带符号整型	32	－2 147 483 648～2 147 483 647	int i＝10；
uint	无符号整型	32	0～4 294 967 295	uint i＝10； uint i＝10U；
long	带符号长整型	64	－9 223 372 036 854 775 808～ 9 223 372 036 854 775 807	long i＝10； long i＝10L；
ulong	无符号长整型	64	0～18 446 744 073 709 551 615	ulong i＝16； ulong i＝16U； ulong i＝16L； ulong i＝16UL；

2. 实数类型

C＃中实数类型包括单精度浮点数、双精度浮点数和固定精度的浮点数。它们的差别主要在于取值范围和精度不同。具体各实数类型及其取值范围与精度如表 6.2 所示。

表 6.2　实数类型及其取值范围与精度

类型标识符	说　明	取 值 范 围	示　例
float	单精度浮点数	$\pm 1.5\times 10^{-45}\sim 3.4\times 10^{38}$，精度为 7 位数	float f=1.23；
double	双精度浮点数	$\pm 5.0\times 10^{-324}\sim 1.7\times 10^{308}$，精度为 15～16 位数	double d=1.23；
decimal	固定精度的浮点数	$1.0\times 10^{-28}\sim 7.9\times 10^{28}$ 的之间，精度为 28～29 位有效数字	decimal d=1.23；

3. 字符类型

在 C# 中，字符类型采用国际上公认 16 位 Unicode 字符集表示形式，用它可以来表示世界上大多种语言。其取值范围为'\u0000'～'\uFFFF'，即 0～65 535。字符类型的标识符是 char，因此也可称为 char 类型。例如，可以采用如下方式字符变量赋值：

```
char c = 'H';                    //字符 H
char c = '\x0048';               //字符 H，十六进制转义符(前缀为\x)
char c = '\u0048';               //字符 H，Unicode 表示形式(前缀为\u)
char c = '\r';                   //回车，转义字符(用于在程序中指代特殊的控制字符)
```

在表示一个字符常数时，单引号内的有效字符数量必须且只能是一个，并且不能是单引号或反斜杠(\)。

4. 布尔类型

布尔类型数据用于表示逻辑真和逻辑假，布尔类型的类型标识符是 bool。

布尔类型常数只有两种值：true(代表“真”)和 false(代表“假”)。布尔类型数据主要应用在流程控制中，往往通过读取或设定布尔类型数据的方式来控制程序的执行方向。

注意：在 C# 语言中，bool 类型不能像 C/C++ 语言那样可能直接转换为 int 类型，例如，int a=(2<3)；在 C/C++ 中都是正确的，但在 C# 中不允许这样，会出现“无法将类型 bool 隐式转换为 int”的编译错误。

6.2.2　引用类型

C# 中的另一大数据类型是引用类型，引用类型也称为参考类型。和值类型相比，引用类型的变量不直接存储所包含的值，而是指向它所要存储的值。换句话说，值类型变量的内存空间中存储的是实际数据，而引用类型变量在其内存空间中存储的是一个指针，该指针指向存储数据的另一块内存位置。由此可见，值类型变量的内存开销要小，访问速度要快，而引用类型变量的内存开销要大，访问速度稍慢。

引用类型共分 4 种类型：类、接口、数组、委托等。

1. object 类

object 是 C# 中所有类型(包括所有的值类型和引用类型)的基类，C# 中的所有类型都直接或间接地从 object 类中继承。因此，对一个 object 的变量可以赋予任何类型的值：

```
object obj1;                     //定义 obj1 对象
obj1 = 1.23;
object obj2 = "China";           //定义 obj2 对象并赋初值
```

2. String 类

C# 还定义了一个 String 类，表示一个 Unicode 字符序列，专门用于对字符串的操作。这个

类是在 .NET Framework 的命名空间 System 中定义的，string 是类 System.String 的别名。

字符串在实际中应用非常广泛，利用 String 类中封装的各种内部操作，可以很容易完成对字符串处理。例如：

```
string str1 = "123" + "abc";        //" + "运算符用于连接字符串
char c = "Hello World!"[1];         //"[]"运算符可以访问 string 中的单个字符,c = 'e'
string str2 = "China";
string str3 = @"China";             //用@表示后跟一个严格字符串,其中\n 等不再作为转义符
bool b = (str2 == str3);            //" == "运算符用于两个字符串比较,b = true
```

6.2.3　变量定义

变量是在程序的运行过程中其值可以发生变化的量，可以在程序中使用变量来存储各种各样的数据，并对它们进行读、写、运算等操作。

注意：不同于 JavaScrip，C#是强类型的语言，在定义变量前必须指定数据类型，而且只能赋给该类型的数据值。

1. 变量定义方法

在 C#程序中使用某个变量之前，必须要告诉编译器它是一个什么样的变量，通过对变量定义来完成。定义变量的方法如下：

```
[访问修饰符]  数据类型  变量名  [ = 初始值];
```

例如：

```
string name = "王华";
int a = 1,b = 2,c = 3;
```

变量具有作用范围，C#是纯面向对象的语言，没有像 C/C++中那样的全局变量，只能在类中定义变量，包括类字段和类函数成员中定义的变量，前者的作用范围是整个类，后者的作用范围是该变量所在的函数成员。

2. 理解值类型的变量

在定义变量时，如果指定变量的类型是值类型，那么这个变量就是值类型的变量（或值变量）。值变量直接把值存放在这个变量名标记的存储位置上，值类型的变量是在栈空间中分配的。

当定义一个值类型变量并且给它赋值时，这个变量只能存储相同类型的数据。所以，一个 int 类型的变量就只能存放 int 类型的数据。另外，当把值赋给某个值类型的变量时，C#首先创建这个值的一个备份，然后把这个备份放在变量名所标记的存储位置上。例如：

```
int x;
int y = 2;
x = y;
```

在这段代码中，当把变量 y 赋值给 x 时，程序会创建变量 y 的值的备份，即 2，然后把这个值放到 x 中，如图 6.1 所示。如果后面的程序修改了 y 的值，就不会影响 x 的值。

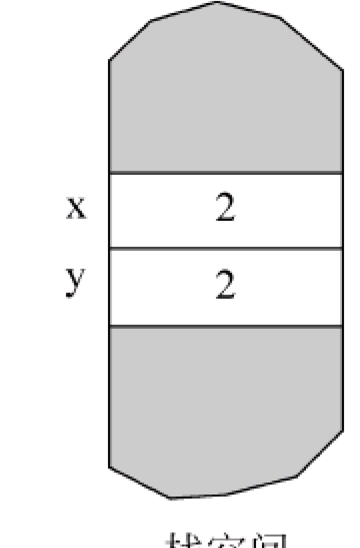


图 6.1　值变量的空间分配

3. 理解引用类型的变量

在定义变量时，如果指定变量的类型是引用类型，那么这个变量就是引用类型的变量（或引用变量）。引用表示所使用的是变量或对象的地址而不是变量或对象本身。当定义引用变量时，程序只是分配了存放这个引用的存储空间，它是在栈空间中分配的。还要创建对象实例并把该实例的存储地址赋给该引用变量，就需要使用 new 操作符。例如：

```
MyClass p1;                    //MyClass 是已声明的类或类型
p1 = new MyClass();
MyClass p2 = p1;
```

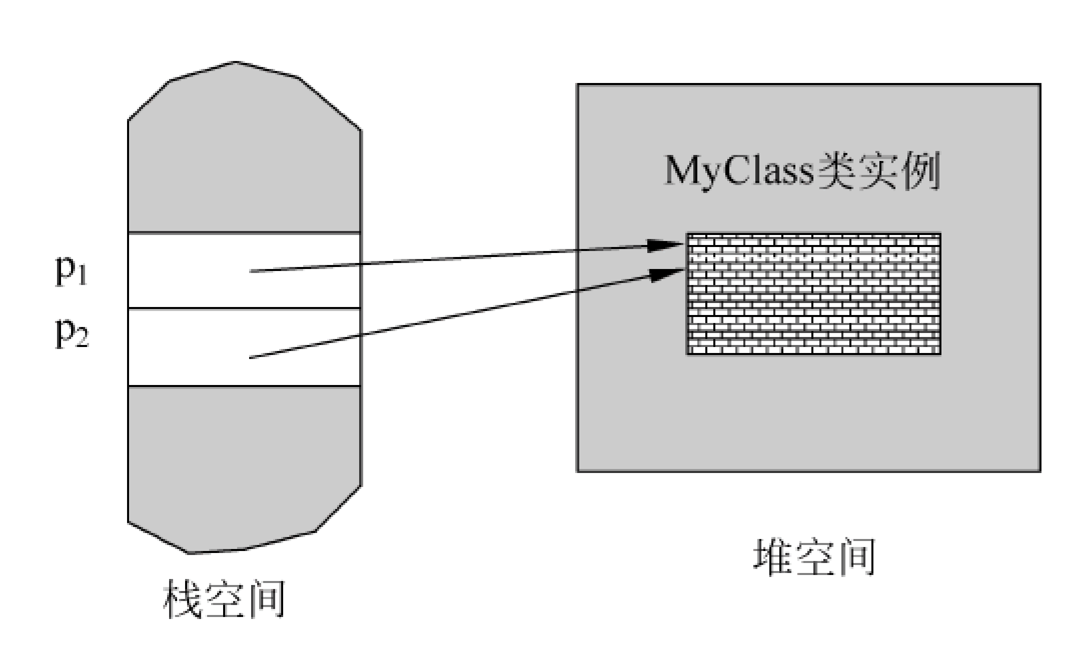


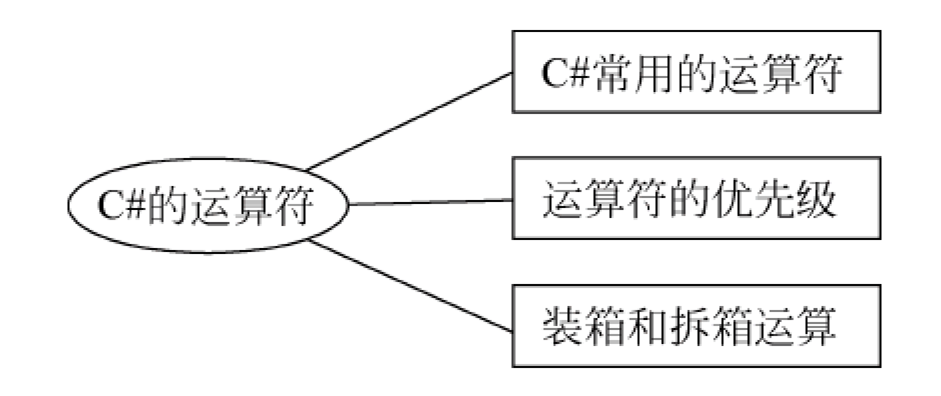
图 6.2　引用类型变量的空间分配

第 1 个语句定义了 MyClass 类的一个引用类型变量 p1。第 2 个语句使用 new 操作符来创建 MyClass 类实例，C＃会在堆存储空间中分配该实例，然后把这个实例的地址赋给这个引用变量 p1，这个引用变量就可以用来引用堆中创建的那个实例了。第 3 个语句定义了 MyClass 类的一个引用类型变量 p2，并让它指向 p1 所指的 MyClass 实例。p1 和 p2 引用变量的空间分配如图 6.2 所示。通常将 p1 和它所指的 MyClass 实例称为 p1 对象。

由于实例是在堆存储空间中分配的，在程序执行时，.NET Framework 会跟踪堆存储空间，一旦某个指向实例的引用变量超出了作用范围，就自动释放该实例，不需要程序员编写专门的代码来释放实例空间，从而达到垃圾自动回收的目的，这是 C＃的优点之一。

6.3　C＃的运算符

知识梳理



6.3.1　C＃常用的运算符

C＃中表达式由运算数和运算符组成，运算符是用来定义类实例中表达式运算数的运算。依照运算符作用的运算数的个数来分，C＃中共有 3 种类型的运算符：

- 一元运算符：一元运算符带一个运算数并使用前缀符（如－x）或后缀符（如 x＋＋）。
- 二元运算符：二元运算符带两个运算数并且全部使用中缀符（如 x ＋ y）。
- 三元运算符：只存在唯一一个三元运算符“?:”。三元运算符带三个运算数并使用中缀符（c? x:y）。

下面分别给出使用运算符的例子：

```
int x = 3;
int y = 6;
int z;
x++;                       //一元运算符
--x;                       //一元运算符
z = x + y;                 //二元运算符
y = (x < 10 ? 0 : 1);      //三元运算符
```

注意：最后一行代码表示当 x＜10 成立时 y 取值为 0，否则取值为 1。

表 6.3 列出了 C#所支持的运算符。

表 6.3　C#中的运算符

运算符类别	运　算　符
算术	+、-、*、/
逻辑	&、|、^、&&、||
字符串串连	+
递增、递减	++、--
移位	<<、>>
关系	==、!=、<=、>=
赋值	=、+=、-=、*=、/=、%=、|=、^=、<<=、>>=
成员访问	.
索引	[]
条件	?:
委托串连和删除	+=、-=
创建对象	new
类型信息	is、as、sizeof、typeof
溢出异常控制	checked、unchecked

6.3.2　运算符的优先级

当一个表达式包含多个运算符时，运算符的优先级控制着单个运算符求值的顺序，一般地，先执行优先级高的运算符，同级的运算符按照从左到右的顺序执行，括号的优先级最高。例如，对于表达式：

```
x + y * z
```

首先求出 y * z，然后将结果与 x 相加，因为 * 的优先级比+的优先级要高。如果需要调整优先级，可以使用括号“()”。例如，需要先求 x 与 y 的和，然后再将结果与 z 相乘，则可以编写如下表达式：

```
(x + y) * z
```

还有考虑运算符的结合性，函数是左结合的，其优先级高。例如，对于表达式：

```
2 * (3 + 5) - f(7)
```

首先计算 f(7)的值，接着计算(3+5)，再计算 2 * (3+5)，最后做减法运算。

表 6.4 总结了常见运算符从高到低的优先级顺序，每个组中的运算符具有相同的优先级。

表 6.4　常见运算符的优先级

<table>
<tr><th>运算符类别</th><th>运　算　符</th><th>运算符类别</th><th>运　算　符</th></tr>
<tr><td>基本</td><td>x.y、f(x)、a[x]、x＋＋、x－－、new、typeof、sizeof、checked、unchecked、-></td><td>逻辑“与”</td><td>x & y</td></tr>
<tr><td>一元</td><td>＋x、－x、!x、～x、＋＋x、－－x、(T)x</td><td>逻辑“异或”</td><td>x ^ y</td></tr>
<tr><td>乘法</td><td>x * y、x / y、x % y</td><td>逻辑“或”</td><td>x || y</td></tr>
<tr><td>加法</td><td>x ＋ y、x － y</td><td>条件“与”</td><td>x && y</td></tr>
<tr><td>移位</td><td>x << y、x >> y</td><td>条件“或”</td><td>x||y</td></tr>
<tr><td>关系</td><td>x > y、x <= y、x >= y</td><td>条件运算</td><td>?:</td></tr>
<tr><td>相等</td><td>x == y、x!= y</td><td>赋值运算</td><td>x = y、x += y、x -= y、x *= y、x/=y、x %= y、x &= y、x|= y、x ^= y、x <<= y、x >>= y</td></tr>
</table>

6.3.3　装箱和拆箱运算

装箱和拆箱运算是 C＃类型系统中重要的概念。通过装箱和拆箱实现值类型和引用类型数据的相互转换，也就是说，装箱和拆箱运算是实现值类型和引用类型相互转换的桥梁。

1．装箱转换

装箱转换是指将一个值类型的数据隐式地转换成一个引用类型的数据。把一个值类型装箱，就是创建一个 object 类型的实例，并把该值类型的值复制给这个 object 实例。

例如，下面的语句就执行了装箱转换：

```
int i = 10;
object obj = i;                //装箱
```

2．拆箱转换

拆箱转换是指将一个引用类型的数据显式地转换成一个值类型数据。

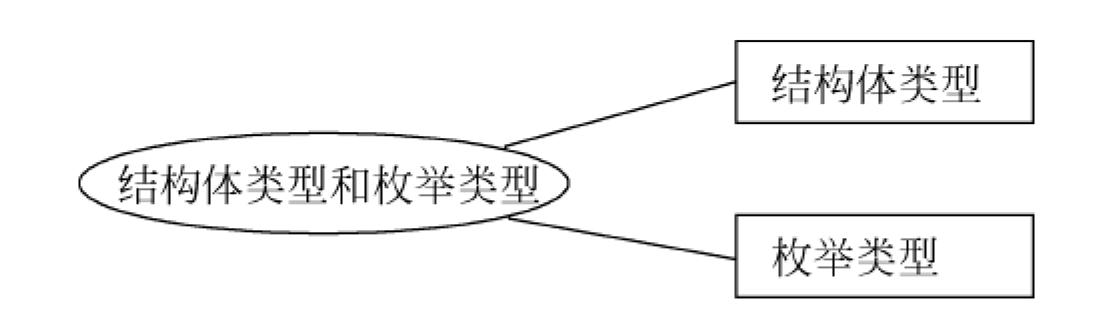
拆箱操作分为两步：首先检查对象实例，确保它是给定值类型的一个装箱值；然后把实例的值复制到值类型数据中。例如，下面的语句就执行了拆箱转换：

```
object obj = 10;
int i = (int)obj;              //拆箱
```

拆箱转换需要(而且必须)执行显式转换，这是它与装箱转换的不同之处。

6.4　结构体类型和枚举类型

知识梳理



6.4.1　结构体类型

结构体类型是一种值类型，对应变量的值保存在栈内存区域。

1. 结构体类型的声明

结构体类型由若干“成员”组成的。数据成员称为字段，每个字段都有自己的数据类型。声明结构体类型的一般格式如下：

```
struct 结构体类型名称
{    [字段访问修饰符] 数据类型 字段 1;
     [字段访问修饰符] 数据类型 字段 2;
     ...
     [字段访问修饰符] 数据类型 字段 n;

};
```

其中，struct 是结构体类型的关键字。“字段访问修饰符”主要取值有 public 和 private(默认值)，public 表示可以通过该类型的变量访问该字段，private 表示不能通过该类型的变量访问该字段。

例如，以下声明一个具有姓名和年龄的结构体类型 Student：

```
struct Student                  //声明结构体类型 Student
{    public int xh;             //学号
     public string xm;          //姓名
     public string xb;          //性别
     public int nl;             //年龄
     public string bh;          //班号
}
```

在上述结构体类型声明中，结构体类型名称为 Student。该结构体类型由 5 个成员组成，第 1 个成员是 xh，为整型变量；第 2 个成员是 xm，为字符串类型；第 3 个成员是 xb，为字符串类型；第 4 个成员是 nl，为整型变量；第 5 个成员是 bh，为字符串类型。

2. 结构体类型变量的定义

声明一个结构体类型后，可以定义该结构体类型的变量(简称为结构体变量)。定义结构体变量的一般格式如下：

```
结构体类型 结构体变量;
```

例如，在前面的结构体类型 Student 声明后，定义它的两个变量如下：

```
Student s1,s2;
```

3. 结构体变量的使用

结构体变量的使用主要包括字段访问和赋值等，这些都是通过结构体变量的字段来实现的。

(1) 访问结构体变量字段

访问结构体变量字段的一般格式如下：

```
结构体变量名.字段名
```

例如,s1.xh 表示结构体变量 s1 的学号,s2.xm 表示结构体变量 s2 的姓名。

(2) 结构体变量的赋值

结构体变量的赋值有两种方式:

- 结构体变量的字段赋值:使用方法与普通变量相同。
- 结构体变量之间赋值:要求赋值的两个结构体变量必须类型相同,例如:

```
s1 = s2;
```

这样 s2 的所有字段值赋给 s1 的对应字段。

6.4.2 枚举类型

枚举类型也是一种自定义数据类型,允许用符号代表数据。枚举是指程序中某个变量具有一组确定的值,通过"枚举"可以将其值一一列出来。这样,使用枚举类型,就可以将常用颜色用符号 Red、Green、Blue、White、Black 来表示,从而提高了程序的可读性。

1. 枚举类型的声明

枚举类型使用 enum 关键字声明,其一般语法形式如下:

```
enum 枚举名 {枚举成员 1,枚举成员 2,…}
```

其中,enum 是枚举体类型的关键字。例如,以下声明一个名称为 color 的表示颜色的枚举类型:

```
enum Color {Red,Green,Blue,White,Black}
```

在声明枚举类型后,可以通过枚举名来访问枚举成员,其使用语法如下:

```
枚举名.枚举成员
```

2. 枚举成员的赋值

在声明的枚举类型中,每一个枚举成员都有一个相对应的常量值,默认情况下 C# 规定第 1 个枚举成员的值取 0,它后面的每一个枚举成员的值按加上 1 递增。例如,前面 Color 中,Red 值为 0,Green 值为 1,Blue 值为 2,依次类推。

可以为一个或多个枚举成员赋整型值,当某个枚举成员赋值后,其后的枚举成员没有赋值的话,它自动在前一个枚举成员值之上加 1。例如:

```
enum Color { Red = 0, Green, Blue = 3, White, Black = 1};
```

则这些枚举成员的值分别为 0、1、3、4、1。

3. 枚举类型变量的定义

声明一个枚举类型后,可以定义该枚举类型的变量(简称为枚举变量)。定义枚举变量的一般格式如下:

```
枚举类型 枚举变量;
```

例如,在前面的枚举类型 Color 声明后,定义它的两个变量如下:

```
Color c1,c2;
```

4. 枚举变量的使用

枚举变量的使用赋值和访问等。

(1) 枚举变量的赋值

枚举变量赋值的语法格式如下：

```
枚举变量 = 枚举名.枚举成员
```

例如：

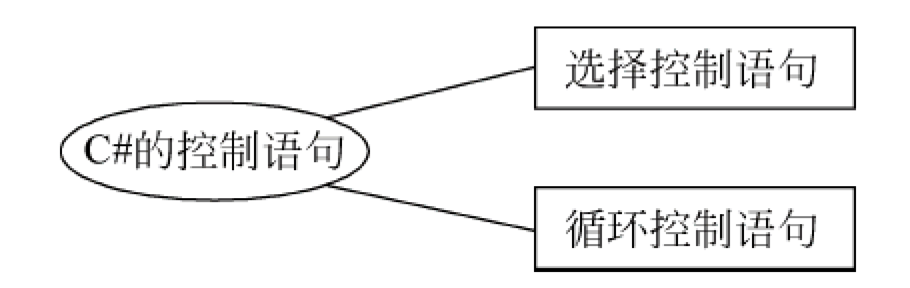
```
c1 = Color.Red;
```

(2) 枚举变量的访问

枚举变量像普通变量一样直接访问。

6.5　C♯的控制语句

知识梳理



6.5.1　选择控制语句

C♯中的选择控制语句有 if 语句、if-else 语句、if-else if 语句和 switch 语句，它们根据指定条件的真假值确定执行哪些简单语句，其中简单语句既可以是单个语句，也可以是用“{}”括起来的复合语句。

1. if 语句

if 语句用于在程序中有条件地执行某一语句序列，其基本语法格式如下：

```
if (条件表达式) 语句;
```

其中，“条件表达式”是一个关系表达式或逻辑表达式，当“条件表达式”为 true 时，执行后面的“语句”。

2. if-else 语句

如果希望 if 语句在“条件表达式”为 true 和 false 时分别执行不同的语句，则用 else 来引入条件表达式为 false 时执行的语句序列，这就是 if-else 语句，它根据不同的条件分别执行不同的语句序列，其语法形式如下：

```
if (条件表达式)
    语句 1;
else
    语句 2;
```

其中的“条件表达式”是一个关系表达式或逻辑表达式。当“条件表达式”为 true 时执行“语句

1”；当“条件表达式”为 false 时执行“语句 2”。

3. if-else if 语句

if-elseif 语句用于进行多重判断，其语法形式如下：

```
if (条件表达式 1) 语句 1;
else if (条件表达式 2) 语句 2;
   ⋮
else if (条件表达式 n) 语句 n;
else  语句 n+1;
```

先计算“条件表达式 1”的值。如果为 true，则执行“语句 1”；如果“条件表达式 1”的值为 false，则继续计算“条件表达式 2”的值。如果为 true，则执行“语句 2”；如果“条件表达式 2”值为 false，则继续计算“条件表达式 3”的值，依此类推。如果所有条件中给出的表达式值都为 false，则执行 else 后面的“语句 n+1”。如果没有 else，则什么也不做，转到该 if-else if 语句后面的语句继续执行。

例如，如下 Button1_Click 事件处理方法就是采用 if-elseif 语句，将用户在文本框 TextBox1 中输入的学生分数转换成等级，并在文本框 TextBox2 中显示出来：

```
protected void Button1_Click(object sender, EventArgs e)
{    int n;
     if (TextBox1.Text!= "")
     {    n = int.Parse(TextBox1.Text);
          if (n >= 90)
               TextBox2.Text = "优秀";
          else if (n >= 80)
           TextBox2.Text = "优良";
          else if (n >= 70)
               TextBox2.Text = "中等";
          else if (n >= 60)
               TextBox2.Text = "及格";
          else
               TextBox2.Text = "不及格";
     }
}
```

说明： C# 和 JavaScript 一样，也是基于事件编程的，通常事件处理方法都有 sender 和 e 两个参数，其中 sender 参数传递引发事件的对象的引用，e 参数传递特定于要处理的事件的对象。

4. switch 语句

switch 语句也称为开关语句，用于有多重选择的场合，用于测试某一个变量具有多个值时所执行的动作。switch 语句的语法形式如下：

```
switch (表达式)
{    case 常量表达式 1: 语句 1;
     case 常量表达式 2: 语句 2;
      ⋮
     case 常量表达式 n: 语句 n;
     default:语句 n+1;
}
```

switch 语句控制传递给与“表达式”值匹配的 case 块。switch 语句可以包括任意数目的 case 块，但是任何两个 case 块都不能具有相同“常量表达式”值。语句体从选定的语句开始执行，直到 break 语句将控制传递到 case 块以外。在每一个 case 块(包括 default 块)的后面，都必须有一个跳转语句(如 break 语句)。C#不支持从一个 case 块显式贯穿到另一个 case 块(这一点与 C++中的 switch 语句不同)。但有一个例外，当 case 语句中没有代码时，可以不包含 break 语句。

如果没有任何 case 表达式与开关值匹配，则控制传递给跟在可选 default 标签后的语句。如果没有 default 标签，则控制传递到 switch 语句以外。

例如，如下 Button1_Click 事件处理方法就是采用 switch 语句，将用户在文本框 TextBox1 中输入的学生分数转换成等级，并在文本框 TextBox2 中显示出来：

```
protected void Button1_Click(object sender, EventArgs e)
{    int n;
     if (TextBox1.Text!= "")
     {    n = int.Parse(TextBox1.Text);
          switch(n/10)              //整除
          {    case 9: TextBox2.Text = "优秀"; break;
               case 8: TextBox2.Text = "优良";break;
               case 7: TextBox2.Text = "中等";break;
               case 6: TextBox2.Text = "及格";break;
               default: TextBox2.Text = "不及格"; break;
          }
     }
}
```

6.5.2　循环控制语句

C#中的循环控制语句有 while、do-while 和 for 语句，另外，break 和 continue 语句用于结束整个循环和结束当前一趟循环。

1. while 语句

while 语句的一般语法格式如下：

```
while (条件表达式)语句;
```

当“条件表达式”的运算结果为 true 时，则重复执行“语句”。每执行一次“语句”后，就会重新计算一次“条件表达式”，当该表达式的值为 false 时，while 循环结束。

例如，如下 Button1_Click 事件处理方法就是采用 while 语句，求 1～n 之和(其中 n 由用户在文本框 TextBox1 中输入)，并将结果在文本框 TextBox2 中显示。

```
protected void Button1_Click(object sender, EventArgs e)
{    int n,i = 1,s = 0;
     if (TextBox1.Text!= "")
     {    n = int.Parse(TextBox1.Text);
          while (i <= n)
          {    s += i;
               i++;
          }
          TextBox2.Text = string.Format("{0}",s);
     }
```

```
}
```

2. do-while 语句

do-while 语句的一般语法格式如下：

```
do
{   语句;
} while (条件表达式);
```

do-while 语句每一次循环执行一次“语句”，然后计算“条件表达式”是否为 true，如果是，则继续执行循环，否则结束循环。与 while 语句不同的是，do-while 循环中的“语句”至少会执行一次，而 while 语句当条件第一次就不满足时，语句一次也不会被执行。

例如，如下 Button1_Click 事件处理方法就是采用 do-while 语句，求 1～n 之和(其中 n 由用户在文本框 TextBox1 中输入)，并将结果在文本框 TextBox2 中显示。

```
protected void Button1_Click(object sender, EventArgs e)
{    int n, i = 1, s = 0;
     if (TextBox1.Text!= "")
     {    n = int.Parse(TextBox1.Text);
          do
          {    s += i;
               i++;
          } while (i <= n);
          TextBox2.Text = string.Format("{0}",s);
     }
}
```

3. for 语句

for 语句通常用于预先知道循环次数的情况，其一般语法格式如下：

```
for (表达式 1;表达式 2;表达式 3) 语句;
```

其中，“表达式 1”可以是一个初始化语句，一般用于对一组变量进行初始化或赋值。“表达式 2”用作循环的条件控制，它是一个条件或逻辑表达式，当其值为 true 时，继续下一次循环；当其值为 false 时，则终止循环。“表达式 3”在每次循环执行完后执行，一般用于改变控制循环的变量。“语句”在“表达式 2”为 true 时执行。具体来说，for 循环的执行过程为：

① 执行“表达式 1”。

② 计算“表达式 2”的。

③ 如果“表达式 2”的值为 true，先执行后面的“语句”，再执行“表达式 3”，然后转向步骤①；如果“表达式 2”的值为 false，则结束整个 for 循环。

例如，如下 Button1_Click 事件处理方法就是采用 for 语句，求 1～n 之和(其中 n 由用户在文本框 TextBox1 中输入)，并将结果在文本框 TextBox2 中显示：

```
protected void Button1_Click(object sender, EventArgs e)
{    int n, i, s = 0;
     if (TextBox1.Text!= "")
     {    n = int.Parse(TextBox1.Text);
          for (i = 1;i <= n;i++)
               s += i;
          TextBox2.Text = string.Format("{0}",s);
```

```
    }
}
```

另外，C♯还提供了 foreach 循环语句，与 for 循环语句类似，用于对容器中的元素进行遍历。其使用语法格式如下：

```
foreach(数据类型 标识符 in 表达式)
    语句;
```

其中，"表达式"指定要遍历的容器，包括 C♯数组，System. Collection 名称空间的集合，以及用户定义的集合等。例如：

```
int[] a = {1,2,3};
foreach(int t in a)              //每执行一次,循环变量 t 依次取集合中的一个元素
{
    //对变量 t 进行读操作
}
```

上述代码的运行效果就是依次遍历数组 a 的所有元素，将数组 a 的各元素依次赋值给变量 t。

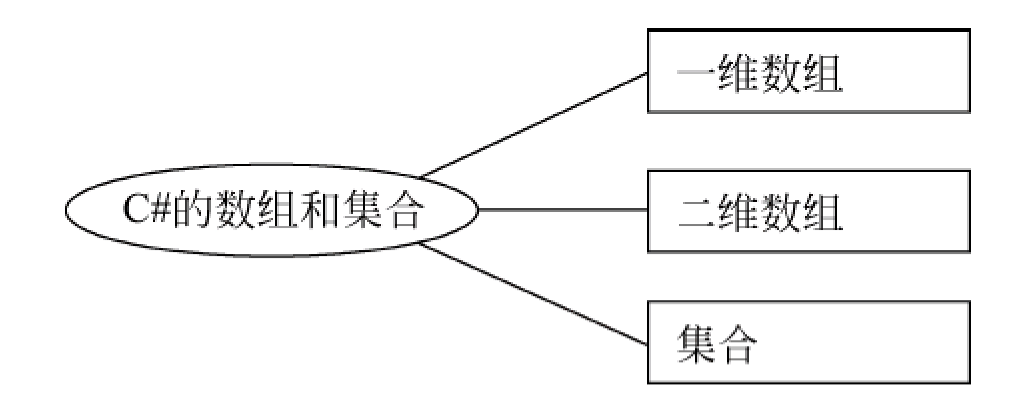
4. break 和 continue 语句

break 语句将使程序从当前的循环语句(do、while 和 for)内跳转出来，接着执行循环语句后面的语句。break 语句还可以用于从 switch 语句中跳出，接着执行 switch 语句后面的语句。

continue 语句也用于循环语句，它类似于 break，但它不是结束整个循环，而是结束循环语句的当前一次循环，接着执行下一次循环。在 while 和 do-while 循环语句中，执行控制权转至对"条件表达式"的继续判断，在 for 语句中，转去执行"表达式 2"。另外，不同于 break 语句，continue 语句不能用于 switch 语句中。

6.6　C♯的数组和集合

知识梳理



6.6.1　一维数组

1. 一维数组的定义

定义一维数组的语法格式如下：

```
数组类型[]  数组名;
```

其中,“数据类型”为 C＃中合法的数据类型;“数组名”为 C＃中合法的标识符。

例如,如下语句定义了双精度数组 a:

```
double[] a;
```

在定义数组后,必须对其进行初始化才能使用。初始化数组有两种方法:动态初始化和静态初始化。

说明:不同于 JavaScript 中的数组,C＃中的数组所有元素的数据类型必须相同。

2. 一维数组的动态初始化

动态初始化需要借助 new 运算符,为数组元素分配内存空间,并为数组元素赋初值,数值类型初始化为 0,布尔类型初始化为 false,字符串类型初始化为 null。

动态初始化数组的格式如下:

```
数组类型[] 数组名 = new 数据类型[n]{元素值0,元素值1, …,元素值n-1};
```

其中,“数组类型”是数组中数据元素的数据类型;n 为“数组长度”,可以是整型常量或变量;后面一层大括号为初始值部分。

3. 一维数组的静态初始化

静态初始化数组时,必须与数组定义结合在一起,否则会出错。静态初始化数组的格式如下:

```
数据类型[]  数组名 = {元素值0,元素值1, …,元素值n-1};
```

例如,以下语句是对整型数组 a 的静态初始化。

```
int[] a = {1,2,3,4,5};
```

4. 访问一维数组中的元素

为了访问一维数组中的某个元素,需指定数组名称和数组中该元素的下标(或索引)。所有元素下标从 0 开始,到数组长度减 1。例如,以下语句输出数组 a 的所有元素值:

```
for (i = 0;i < 5;i++)
    //输出 a[i]的值;
```

6.6.2 二维数组

1. 二维数组的定义

定义二维数组的语法格式如下:

```
数组类型[,]  数组名;
```

其中,“数据类型”为 C＃中合法的数据类型;“数组名”为 C＃中合法的标识符。

例如,以下语句定义了 3 个二维数组,即整型数组 x、双精度数组 y 和字符串数组 z。

```
int[,] x;
double[,] y;
string[,] z;
```

对于多维数组,可以作类似的推广。例如,以下语句定义了一个三维数组 p。

```
int[,,] p;
```

2. 二维数组的动态初始化

动态初始化二维数组的格式如下：

```
数据类型[,] 数组名 = new 数据类型[m][n]{{元素值0,0,元素值0,1,…,元素值0,n-1},
                                     {元素值1,0,元素值1,1,…,元素值1,n-1},
                                      …
                                     {元素值m-1,0,元素值m-1,1,…,元素值m-1,n-1}};
```

其中,“数组类型”是数组中数据元素的数据类型；m、n 分别为行数和列数。即各维的长度,可以是整型常量或变量；后面两层大括号中为初始值部分。

(1) 不给定初始值的情况

如果不给出初始值部分,各元素取默认值。例如：

```
int[,] a = new int[2,3];
```

该数组各数组元素均取默认值 0。

(2) 给定初始值的情况

如果给出初始值部分,各元素取相应的初值,而且给出的初值个数与对应的“数组长度”相等。此时可以省略“数组长度”,因为后面的大括号中已列出了数组中的全部元素。例如：

```
int[,] a = new int[2,3]{{1,2,3},{4,5,6}};
```

或

```
int[,] a = new int[,]{{1,2,3},{4,5,6}};
```

3. 二维数组的静态初始化

静态初始化数组时,必须与数组定义结合,否则会出错。静态初始化数组的格式如下：

```
数据类型[,]  数组名 = {{元素值0,0,元素值0,1,…,元素值0,n-1},
                    {元素值1,0,元素值1,1,…,元素值1,n-1},
                     …
                    {元素值m-1,0,元素值m-1,1,…,元素值m-1,n-1}};
```

例如,以下语句是对整型数组 b 的静态初始化。

```
int[,] b = {{1,2,3},{4,5,6}};
```

4. 访问二维数组中的元素

为了访问二维数组中的某个元素,需指定数组名称和数组中该元素的行下标和列下标。例如,以下语句输出数组 b 的所有元素值。

```
for (i = 0;i < 2;i++)
    for (j = 0;j < 3;j++
        //输出 b[i,j]的值
```

对于多维数组,也可以使用 foreach 语句来循环访问每一个元素。例如：

```
int[,] c = new int[3, 2]{{1,2},{3,4},{5,6}};
foreach (int i in c)
```

```
    //输出 i 的值
```

除了一维数组和二维数组外，C#还提供了多维数组和交错数组等。C#中的数组都是 Array 类对象，该类提供一些方法，用于创建、处理、搜索数组并对数组进行排序，从而充当所有数组的基类。

6.6.3　集合

在定义数组时要指定数组大小，如果需要动态地增加数组大小，可以使用 ArrayList 类，ArrayList 类对象也称为集合，能够实现数组的复杂运算，位于 System. Collections 命名空间。

ArrayList 类的常见属性和方法如表 6.5 所示。

表 6.5　ArrayList 类的常用属性和方法

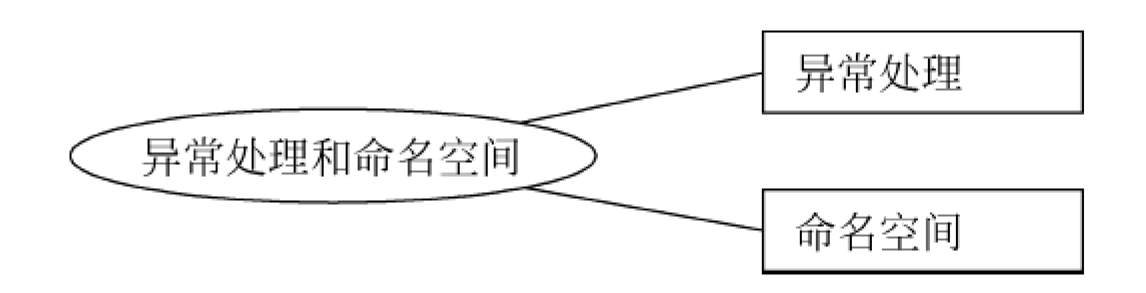
成员类型	名　　称	说　　明
属性	Item	获取或设置位于指定索引处的元素
	Count	获取 ArrayList 中实际包含的元素个数
方法	Add	将对象添加到 ArrayList 的结尾处
	Insert	将元素插入 ArrayList 的指定索引处
	Remove	从 ArrayList 中移除特定对象的第一个匹配项
	Sort()	对整个 ArrayList 中的元素进行排序
	Sort(IComparer)	使用指定的比较器对整个 ArrayList 中的元素进行排序

例如，如下代码创建一个 ArrayList 类对象 myset，添加 4 个元素，并通过 foreach 语句遍历 myset 中的所有元素：

```
ArrayList myset = new ArrayList();
myset.Add(1);
myset.Add("Mary");
myset.Add("2");
myset.Add("Smith");
foreach (var item in myset)
    //输出 item 元素
```

6.7　异常处理和命名空间

知识梳理



6.7.1　异常处理

为了保证程序更加完备，经常在程序中会使用到异常处理语句 try-catch-finally，其使用语法格式如下：

```
try
{   被保护的语句块; }
catch(异常对象声明)
{   捕获到异常时执行的语句块; }
finally
{   完成善后工作的语句块; }
```

其中,各部分的说明如下:

- try 块:封装了程序要执行的代码,如果只这段代码的过程中出现错误或异常情况,就会抛出一个异常。
- catch 块:在 try 块的后面,封装了处理在 try 代码块中出现错误所采取的措施。可以有多个 catch 块用来捕获不同类型的异常。
- finally 块:该块是可选的,如果有,它放在 catch 之后,无论 try 块是否有异常,这个块中的代码都要执行。另外,不能跳出 finally 块,如果采用跳转语句要跳出 try 块,仍要执行 finally 块。

【练一练】 创建好本章的文件系统空网站 CH6,设计一个检测两个整数除法运算错误的网页 webform1.aspx,做除法运算的两个正整数由用户输入。

其设计步骤如下:

① 打开 CH6 网站,选择"网站|添加新项"菜单命令,出现"添加新项-CH6"对话框,在中间列表中选择"Web 窗体",将文件名称改为 webform1.aspx,其他保持默认项(采用代码隐藏模型),单击"添加"按钮。

② 该网页的设计界面如图 6.3 所示,其中其中包含两个 HTML 文字,一个标签 Label1、两个文本框(ID 为 TextBox1~TextBox2)和一个命令按钮 Button1。



图 6.3　webform1 网页设计界面

注意:在 ASP .NET 网页界面设计过程中,可以采用前面几章介绍的 HTML 和 CSS 进行界面美化设计。

③ 双击 Button1 控件,出现代码编辑窗口,输入以下事件过程代码:

```
protected void Button1_Click(object sender, EventArgs e)
{   int a, b, c;
    string mystr = "";
    try
    {   a = int.Parse(TextBox1.Text);
        b = int.Parse(TextBox2.Text);
        c = a/b;
        mystr = string.Format("{0}",c);
    }
    catch(DivideByZeroException ex)
    {  mystr = ex.Message;  }
    finally
    {  Label1.Text = "结果:" + mystr;  }
}
```

其中,DivideByZeroException 是除零异常类,Exception 是异常类,前者是从后者派生的。Exception 类包含 Message 属性,在出现异常时包含相应的错误信息。

注意:C#脚本代码是在服务器端执行的,可以访问服务器控件的内容,而本网页中的

TextBox1、TextBox2 和 Label1 等控件都有 runat＝"server"属性设置，表示它们都是服务器控件，所以可以直接访问。如果使用 JavaScript 脚本代码，由于 JavaScript 脚本代码是在客户端执行的，因此不能直接访问这些服务器控件的内容。这就是 C＃脚本代码和 JavaScript 脚本代码的区别之一。

④ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，在“被除数”文本框中输入 100，在“除数”文本框中输入 20，单击“确定”按钮，没有出现异常，mystr 中保存 5，其结果如图 6.4 所示。

若在“被除数”文本框中输入 100，在“除数”文本框中输入 0，单击“确定”按钮，会出现异常，被 catch 子句检测到，会修改 mystr 的值(此时为“尝试除以零”)，其结果如图 6.5 所示。从中看到，不论是否出现异常，都会执行 finally 中包含的语句。

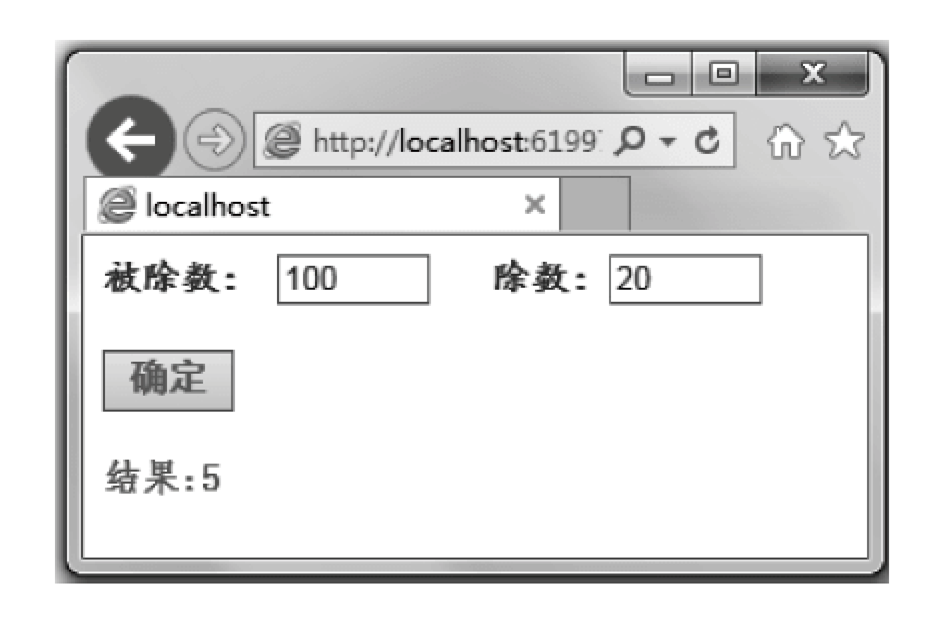


图 6.4　webform1 网页执行界面一



图 6.5　webform1 网页执行界面二

6.7.2　命名空间

在 C＃编程中总会使用到 .NET 类库，它是一个含有上千个类、接口和值类型的库，提供了对系统功能的访问，是创建 Web 应用程序等的基础。其中所有的类按逻辑关系进行分类，也就是命名空间，命名空间提供了一种组织相关类和其他类型的方式。与文件或组件不同，命名空间是一种逻辑组合，而不是物理组合。在 C＃文件中定义类时，可以把它包括在命名空间定义中。以后在定义另一个类，在另一个文件中执行相关操作时，就可以在同一个命名空间中包含它，创建一个逻辑组合，告诉使用类的其他开发人员这两个类是如何相关的以及如何使用它们。

使用命名空间有两种方式，一种是使用别名指令为命名空间定义别名，此后的程序语句就可以使用这个别名来代替定义的这个命名空间；另一种是通过 using 关键字引用命名空间，把该命名空间中的类型导入到包含这个 using 语句的命名空间中，这样就可以直接使用命名空间中的类型的名称。例如：

```
using System;
```

这个语句就是引用 System 命名空间。这样在程序中可以使用 System 命名空间中包含的类或结构体等。在前面所有示例中，进入代码编辑窗口时，其开头部分默认包含如下语句：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

这是引用 .NET Framework 的命名空间，其说明如下：

- System：提供基本类。
- System. Collections. Generic：提供集合类。
- System. Linq：提供用于 LINQ 的类。
- System. Web：提供使浏览器与服务器相互通信的类和接口。
- System. Web. UI：提供用于创建 Web 应用程序用户界面的类和接口。
- System. Web. UI. WebControls：提供在 Web 窗体上创建 Web 服务器控件的类。

其实上述引用并非所有的网页都是需要的，对于这些网页的设计，只需要引用 System 命名空间就可以了。开发人员可以根据需要添加或修改。

在网页代码文件.aspx.cs 中，在任何空白处右击，在弹出的快捷菜单中选择“组织 using|移去未使用的 using”命令，如图 6.6 所示，即可删除网页中没有使用的命名空间的引用语句。

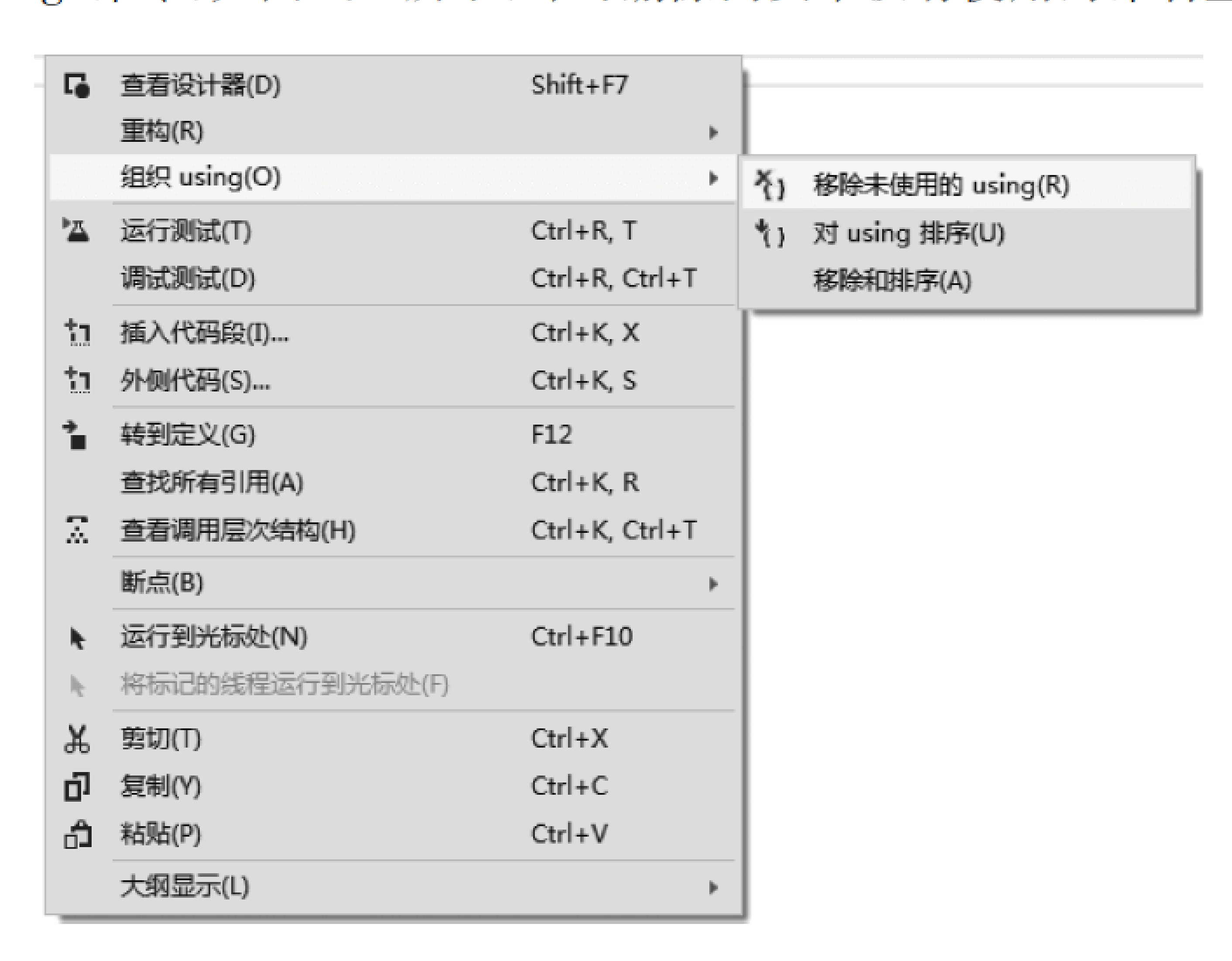
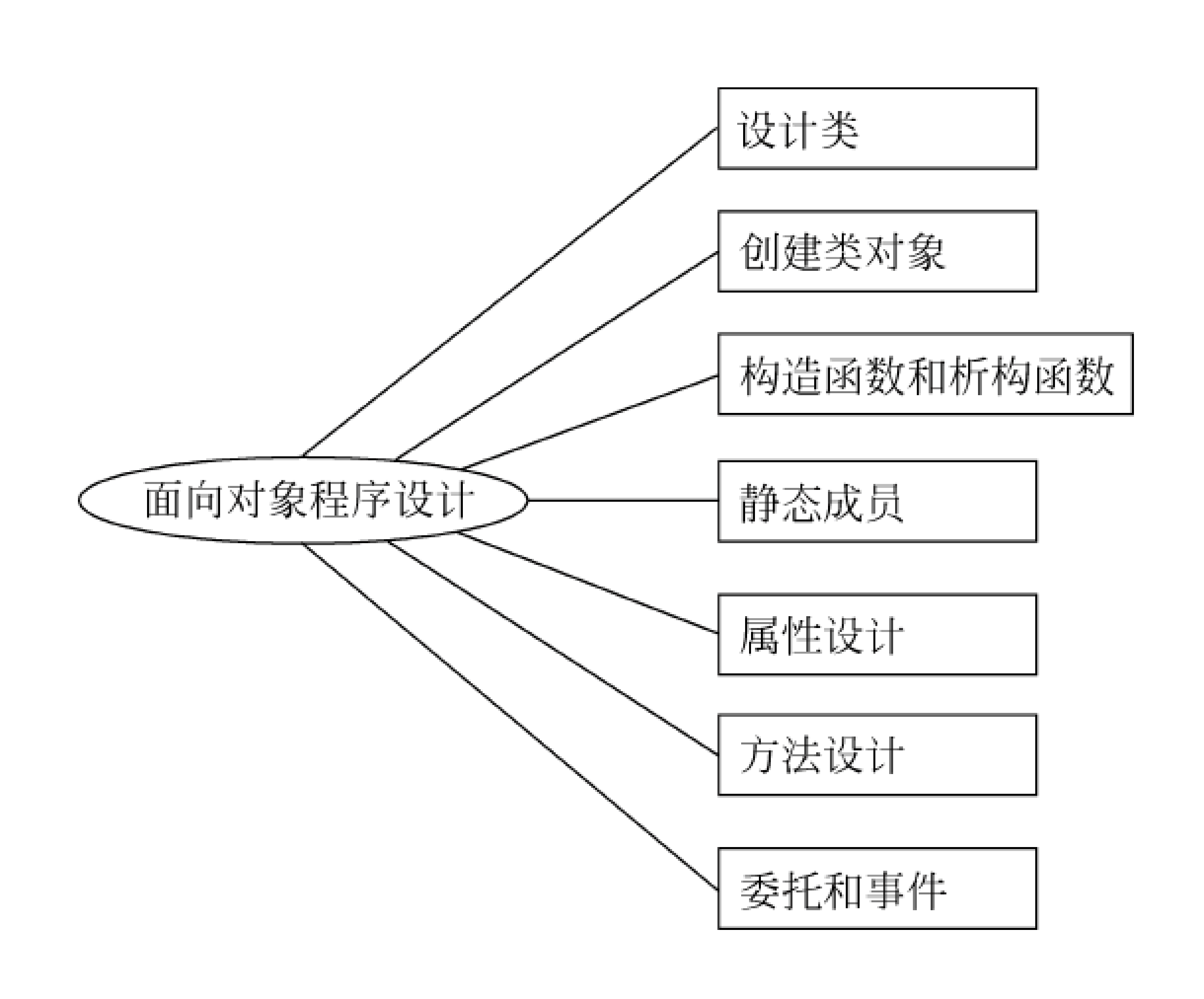


图 6.6　选择“组织 using|移去未使用的 using”命令

6.8　面向对象程序设计

知识梳理



6.8.1　设计类

从计算机语言角度来说，类是一种数据类型，包含数据成员和操作数据成员的方法，而对象是具有这种类型的变量。

1. 类的声明

类的声明语法格式如下：

```
[类的修饰符] class 类名 [:基类名]
{
    //类的成员;
}[;]
```

其中，class 是声明类的关键字；“类名”必须是合法的 C＃标识符；“类的修饰符”有多种，用于指定类的访问级别，其说明如表 6.6 所示。

表 6.6　类的访问修饰符

类的修饰符	说　　明
public	公有类，表示不受限制对该类的访问
protected	保护类，表示只能从所在类和所在类派生的子类进行访问
internal	内部类，只有其所在类才能访问
private	私有类，只有该类才能访问
abstract	抽象类，表示该类是一个不完整的类，不允许建立类的实例
sealed	密封类，不允许从该类派生新的类

2. 类的成员

类的成员可以是类本身所声明的以及从基类中继承而来的。类的成员如表 6.7 所示，总体上分为两大类：字段和属性合起来称为数据成员，主要用于存储类的数据；其他合起来称为函数成员，主要用于数据的处理或操作。

表 6.7　类的成员

类的成员	说　　明
字段	字段存储类要满足其设计所需要的数据，也称为数据成员
属性	属性是类中可以像类中的字段一样访问的方法。属性可以为类字段提供保护，避免字段在对象不知道的情况下被更改
方法	方法定义类可以执行的操作。方法可以接受提供输入数据的参数，并且可以通过参返回输出数据。方法还可以不使用参数而直接返回值
委托	委托定义了方法的类型，使得可以将方法当作另一个方法的参数来进行传递，这种将方法动态地赋给参数的做法，使得程序具有更好的可扩展性
事件	事件是向其他对象提供有关事件发生(如单击按钮或执行某个方法)通知的一种方式
索引器	索引器允许以类似于数组的方式为对象建立索引
运算符	运算符是对操作数执行运算的术语或符号，如＋、＊、<等
构造函数	构造函数是在第一次创建对象时调用的方法。它们通常用于初始化对象的数据
析构函数	析构函数是当对象即将从内存中移除时由运行库执行引擎调用的方法。它们通常用来确保需要释放的所有资源都得到了适当的处理

类的成员也可以使用不同的访问修饰符，从而定义它们的访问级别，类成员的访问修饰符及其说明如表 6.8 所示。

表 6.8　类成员的访问修饰符

类成员的修饰符	说　明
public	公有成员，提供了类的外部界面，允许类的使用者从外部进行访问，这是限制最少的一种访问方式
private	私有成员(默认的)，仅限于类中的成员可以访问，从类的外部访问私有成员是不合法的，如果在声明中没有出现成员的访问修饰符，按照默认方式成员为私有的
protected	保护成员，这类成员不允许外部访问，但允许其派生类成员访问
internal	内部成员，允许同一个命名空间中的类访问
readonly	只读成员，这类成员的值只能读，不能写。也就是说，除了赋予初始值外，在程序的任何一个部分将无法更改这个成员的值

例如，以下声明了一个 Person 类：

```
public class Person                              //声明 Person 类,其访问级别为 public
{    public int pno;                             //编号,为公有字段
     string pname;                               //姓名,为私有字段
     public void setdata(int no,string name)     //定义 setdata 方法
     {   pno = no; pname = name;   }
     public void dispdata()                      //定义 dispdata 方法
     {   Console.WriteLine("{0} {1}", pno, pname);   }
}
```

上述 Person 类的修饰符为 public，为公有类。其中有两个字段，pno 是公有字段；而 pname 是私有字段；另外有两个共有方法。

C# 中类声明和 C++有一个明显的差别是字段可以赋初值。例如，在上面 Person 类声明中可以将 pno 字段改为：

```
public int pno = 101;
```

这样 Person 类的每个对象的 pno 字段都有默认值 101。

3. 分部类

分部类可以将类(结构体或接口等)的声明拆分到两个或多个源文件中。若要拆分类的代码，被拆分类每一部分的定义前边都要用 partial 关键字修饰。分部类的每一部分都可以存放在不同的文件中，编译时会将所有部分组合起来构成一个完整的类声明。

每个网页的逻辑代码中都声明了一个分部类。例如，webform1 网页的逻辑代码 webform1.aspx.cs 中有以下代码：

```
public partial class webform1 : System.Web.UI.Page
{
     ⋮
}
```

表示 webform1 类是一个分部类，它从 System.Web.UI.Page 类派生的。实际上所有网页类都是从 System.Web.UI.Page 类继承的，ASP .NET 将动态编译网页，并在用户第一次请求时运行网页，如果网页发生更改，编译器将自动对该网页进行重新编译。

6.8.2 创建类对象

类和对象是不同的概念。类定义对象的类型，但它不是对象本身。对象是基于类的具体实体，有时称为类的实例。只有定义类对象时才会给对象分配相应的内存空间。

1. 定义类的对象

一旦声明了一个类，就可以用它作为数据类型来定义类对象(简称为对象)。定义类的对象分两步：

(1) 定义对象引用

其语法格式如下：

```
类名 对象名;
```

例如，"Person p;"语句定义Person类的对象引用p。

(2) 创建类的实例

其语法格式如下：

```
对象名 = new 类名( );
```

例如，"p=new Person();"语句创建Person类的对象实例。

以上两步也可以合并成一步。其语法格式如下：

```
类名 对象名 = new 类名();
```

例如，"Person p=new Person();"。

通常将对象引用和对象实例混用，但读者应了解它们之间的差异。上述语句中Person()部分是创建类的实例，然后传递回对该对象的引用并赋给p，这样就可以通过对象引用p操作该对象实例。两个对象引用可以引用同一个对象。例如：

```
Person p1 = new Person();
Person p2 = p1;
```

上述代码中，先创建一个Person对象引用p1，再创建一个Person类实例，由p1指向该实例，在执行p2=p1后，两个对象引用p1、p2都同一个实例。也就是说，可以通过p1和p2操作同一个实例。

对于类中的字段，在所属的类创建对象时被分配存储空间，此时也称字段拥有了自己的实例。如果没有在对象创建表达式中为字段赋值，那么每个字段的初始值都是其类型的默认值。对于所有整数类型和实数类型，以及由整数类型衍生出的字符类型和枚举类型，其默认值为0；对于布尔类型，其默认值为false；而对于所有引用类型，其默认值为null。

2. 访问对象的字段

访问对象字段的语法格式如下：

```
对象名.字段名
```

其中，"."是一个运算符，该运算符的功能是表示对象的成员。

例如，前面定义的p对象的成员变量表示为p.pno。

实际上，通过对象名只能访问类的公有成员，不能访问类的私有成员或保护成员。由于

Person 类的 pname 字段默认是私有的，所以不能使用 p.pname 访问 pname 字段。

3. 调用对象的方法

调用对象的方法的语法格式如下：

```
对象名.方法名(参数表)
```

例如，调用前面定义的 p 对象的成员方法 setdata 为 p.setdata(101,"Mary")。

【练一练】 在 CH6 网站中设计一个网页 webform2.aspx，说明创建类的过程。

其设计步骤如下：

① 打开 CH6 网站，选择"网站|添加新项"菜单命令，出现"添加新项-CH6"对话框，在中间列表中选择"Web 窗体"，将文件名称改为 webform2.aspx，其他保持默认项，单击"添加"按钮，创建一个空的网页。

② 选择"网站|添加新项"菜单命令，出现"添加新项"对话框，从模板列表中选择"类"选项，保持默认类文件名 Class1.cs，如图 6.7 所示。单击"添加"按钮，此时出现如图 6.8 所示的系统消息框，单击"是"按钮，表示将该类文件放在系统自动创建的 App_Code 目录中。



图 6.7　"添加新项"对话框



图 6.8　系统消息框

在 Class1 类文件中设计 Comp 类的代码如下：

```
public class Comp
```

```
{   public int fun1(int n)
    {
        return (n + 1) * n/2;
    }
    public int fun2(int n)
    {   int i,j,k,total = 0;
        for (i = 1; i <= n; i++)
        {   k = 1;
            for (j = 1; j <= i; j++)
                k * = j;
            total += k;
        }
        return total;
    }
}
```

该类没有包含字段，只有两个公有方法，分别计算 1＋2＋…＋n 和 1!＋2!＋…＋n!(n＞0)的值。

③ 保存该文件后，设计 webform2 网页的界面如图 6.9 所示，其中包含一个文本框 TextBox1、两个命令按钮(Button1 和 Button2)和一个标签 Label1。并设置相应的字体和颜色属性。

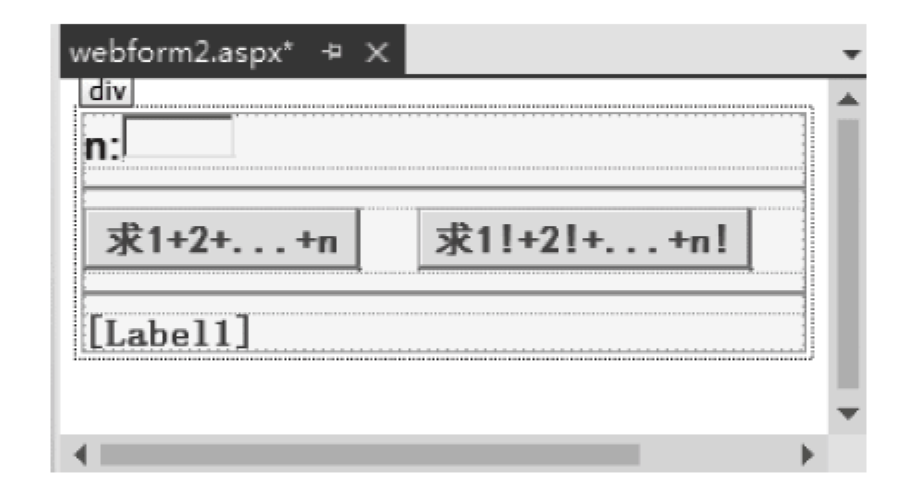


图 6.9　webform2 网页的设计界面

④ 双击 Button1 控件，出现代码编辑窗口，输入以下事件过程代码：

```
protected void Button1_Click(object sender, EventArgs e)
{   Comp myobj = new Comp();
    int n = int.Parse(TextBox1.Text);
    Label1.Text = "1 + 2 + ... + n 的结果: " + myobj.fun1(n);
}
```

⑤ 双击 Button2 控件，出现代码编辑窗口，输入以下事件过程代码：

```
protected void Button2_Click(object sender, EventArgs e)
{   Comp myobj = new Comp();
    int n = int.Parse(TextBox1.Text);
    Label1.Text = "1! + 2! + ... + n! 的结果: " + myobj.fun2(n);
}
```

上述两个事件处理方法相似，都是创建 Class1 类文件中 Comp 类的对象 myobj，然后调用其方法，将结果显示在 Label1 标签中。

⑥ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，输入 n 值为 5，单击“求 1＋2＋…＋n”按钮，其执行结果如图 6.10 所示，再单击“求 1!＋2!＋…＋n!”按钮，其执行结果如图 6.11 所示。

需要说明的是，如果删除 Comp 类的 public 访问修饰符(默认为 private)，执行网页时出现“Comp 不可访问”的错误，因为它受保护级别限制。如果将 public 改为 protected 访问修饰符，执行网页时出现“在命名空间中定义的元素无法显式地声明为 private、protected 或 protected internal”的错误。所以，在类文件中声明的类，如果需要在网页中创建它的对象，需要声明为 public。




图 6.10　webform2 网页执行界面一



图 6.11　webform2 网页执行界面二

如果删除 Comp 类中 fun1 方法的 public 访问修饰符，执行网页时出现“Comp 不包含 fun1 的定义”的错误。所以，对于一个类中成员，若要通过其对象使用它，需要将其设计为 public。

6.8.3　构造函数和析构函数

类的构造函数和析构函数都是类的成员方法，但它们有其特殊性。

1. 构造函数

构造函数是在创建给定类型的对象时执行的类方法。构造函数具有如下性质：

- 构造函数的名称与类的名称相同；
- 构造函数尽管是一个函数，但没有任何类型，它既不属于返回值函数也不属于 void 函数；
- 一个类可以有多个构造函数，但所有构造函数的名称都必须相同，它们的参数各不相同，即构造函数可以重载；
- 当类对象创建时，构造函数会自动地执行；由于它们没有返回类型，不能像其他函数那样进行调用；
- 当类对象声明时，调用哪一个构造函数取决于传递给它的参数类型；
- 构造函数不能被继承。

当定义类对象时，构造函数会自动执行。因为一个类可能会有包括默认构造函数在内的不止一种构造函数，下面讨论如何调用特定的构造函数。

(1) 调用默认构造函数

不带参数的构造函数称为“默认构造函数”。无论何时，只要使用 new 运算符实例化对象，并且不为 new 提供任何参数，就会调用默认构造函数。假设一个类包含默认构造函数，调用默认构造函数的语法如下：

```
类名 对象名 = new 类名();
```

如果没有为对象提供构造函数，则默认情况下 C#将创建一个构造函数，该构造函数实例化对象，并将所有成员变量设置为相应的默认值。

(2) 调用带参数的构造函数

假设一个类中包含带参数的构造函数，调用这种带参数的构造函数的语法如下：

```
类名 对象名 = new  类名(参数表);
```

其中，“参数表”中的参数可以是变量，也可以是表达式。

【练一练】 在 CH6 网站中设计一个网页 webform3.aspx，说明构造函数的使用方法。

其设计步骤如下：

① 打开 CH6 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH6”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform3.aspx，其他保持默认项，单击“添加”按钮，创建一个空的网页。

② 打开 Class1.css 类文件，添加如下代码：

```
public class Comp1
{   int m;
    public Comp1()                                    //默认构造函数
    {  m = 2;  }
    public Comp1(string n)                            //带参数构造函数
    {  this.m = int.Parse(n); }
    public int fun()                                  //求 m!
    {   int s = 1, i;
        for (i = 1; i <= m; i++)
            s *= i;
        return s;
    }
    public int getm()                                 //返回 m 的值
    {  return m;  }
}
```

③ 在本网页中添加一个 HTML 文字。一个文本框 TextBox1、一个命令按钮 Button1 和一个标签 Label1。设计事件处理方法如下：

```
protected void Button1_Click(object sender, EventArgs e)
{   if (TextBox1.Text.Trim() == "")
    {   Comp1 myobj = new Comp1();                    //调用默认构造函数
        Label1.Text = myobj.getm() + "!= " + myobj.fun().ToString();
    }
    else
    {   Comp1 myobj1 = new Comp1(TextBox1.Text.Trim());  //调用带参数构造函数
        Label1.Text = myobj1.getm() + "!= " + myobj1.fun().ToString();
    }
}
```

④ 单击工具栏中的 ▶ Internet Explorer 按钮运行本网页，直接单击“n!”按钮，其运行结果如图 6.12 所示，此时调用 myobj 对象的默认构造函数，求 2!。在文本框中输入 5，再单击“n!”按钮，其运行结果如图 6.13 所示，此时调用 myobj1 对象的带参数构造函数，求 5!。

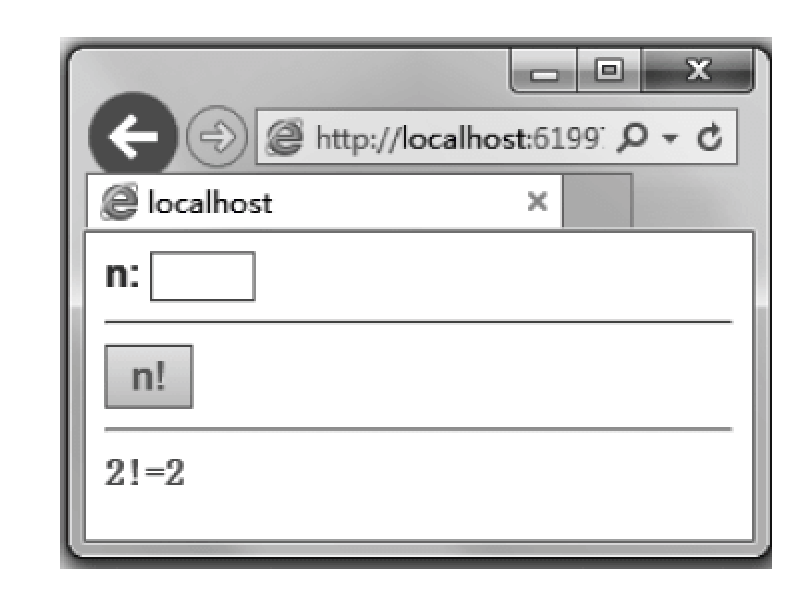


图 6.12　webform3 网页执行界面一

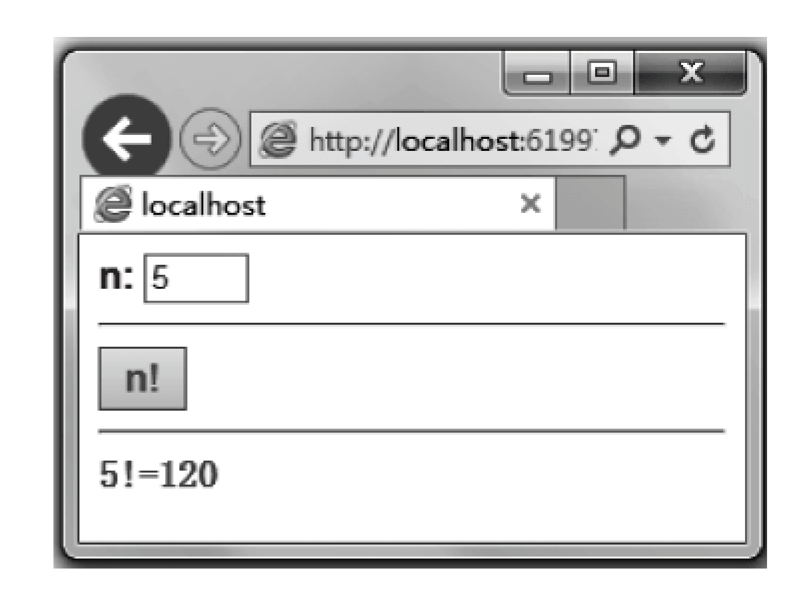


图 6.13　webform3 网页执行界面二

2. 析构函数

在不再需要对象时，希望确保它所占的存储能被收回。C#中提供了析构函数用于专门释放被占用的系统资源。析构函数具有如下性质：

- 析构函数在类对象销毁时自动执行；
- 一个类只能有一个析构函数，而且析构函数没有参数，即析构函数不能重载；
- 析构函数的名称是"～"加上类的名称（中间没有空格）；
- 与构造函数一样，析构函数也没有返回类型；
- 析构函数不能被继承。

6.8.4 静态成员

静态成员主要包括静态字段和静态方法。静态成员属于类所有，而非静态成员属于类的对象所有，所以静态成员也称为类成员，非静态成员也称为对象成员。提出静态成员的目的是为了解决数据共享的问题。

说明：静态成员是属于整个类的，不针对该类的某个对象，称为类成员，所以静态方法是通过类名来调用的。

1. 静态字段

静态字段是类中所有对象共享的成员，而不是某个对象的成员，也就是说静态字段的存储空间不是放在每个对象中，而是和方法一样放在类公共区中。

对静态字段的操作和一般字段一样，定义为私有的静态字段不能从外部访问。静态字段的定义和访问方式如下：

- 静态字段的定义与一般字段相似，但前面要加上 static 关键词；
- 在外部访问静态字段时采用"类名.静态字段名"格式。

例如，有如下类 D：

```
class D
{   int f1;                          //非静态字段
    static int f2;                   //静态字段
    public D() { }
    public D(int a, int b)
    {  f1 = a; f2 = b; }
    public void disp()
    {    //输出 f1 和 f2 的值;  }
};
```

执行以下语句：

```
D d1 = new D(2,6);
D d2 = new D(5,10);
d1.disp();                           //输出 f1 为 2,f2 为 10
d2.disp();                           //输出 f1 为 5,f2 为 10
```

其中类 D 的 f2 字段是静态字段，它由类 D 的所有对象共享，而 f1 是实例字段，每个对象都有自己的副本。

静态成员的生存期和实例成员不同。实例成员在实例创建后才产生，在实例销毁后实例成员也就不存在了。但是即使没有实例，也存在类的静态成员，并且可以访问它们，如可以对

静态字段初始化。

2. 静态方法

静态方法与静态字段类似，都是类的静态成员，独立于类的实例。只要类存在，静态方法就可以使用，静态方法的定义是在一般方法定义前加上 static 关键字。调用静态方法的格式如下：

```
类名.静态方法名(参数表);
```

注意：静态方法只能访问静态字段、其他静态方法和类以外的方法及数据，不能访问类中的非静态成员(因为非静态成员只有对象存在时才有意义)。但静态字段和静态方法可由任意访问权限许可的成员访问。

6.8.5 属性设计

属性描述了对象的具体特性，提供了对类或对象成员的访问。C#中的属性更充分地体现了对象的封装性，不直接操作类的字段，而是通过访问器进行访问。

属性在类模块里是采用如下方式进行声明的，即指定变量的访问级别、属性的类型、属性的名称，然后是 get 访问器或 set 访问器代码块。其语法格式如下：

```
修饰符 数据类型 属性名称
{    get 访问器                    //设置该属性是可读的
     set 访问器                    //设置该属性是可写的
}
```

属性是通过“访问器”来实现的：访问器是数据字段赋值和检索其值的特殊方法。使用 set 访问器可以为数据字段赋值，使用 get 访问器可以检索数据字段的值。

属性是为了保护类的字段。通常的情况下，将字段设计为私有的，设计一个对其进行读或写的属性。在属性的 get 访问器中，用 return 来返回该字段的值，在属性的 set 访问器中可以使用一个特殊的隐含参数 value，该参数包含用户指定的值。

例如，如下 MyClass 类中定义了两个私有字段 xh 和 xm，分别设计了读写它们的 pxh 和 pxm 属性：

```
public class MyClass
{    int xh;
     string xm;
     public int pxh                    //属性 pxh
     {    get   {  return xh; }
          set {   xh = value; }
     }
     public string pxm                 //属性 pxm
     {    get { return xm; }
          set {   xm = value; }
     }
}
```

这样就可以通过属性对相应的字段进行操作。例如：

```
MyClass obj = new MyClass();
obj.pxh = 101; obj.pxm = "王华";
```

这样，obj 的私有字段 xh 值为 101，私有字段 xm 值为“王华”。也可以将属性和对应的字段合二为一。例如，上述 MyClass 可以等价地声明为：

```
public class MyClass
{   public int pxh { set; get; }          //属性 pxh
    public string pxm {set; get; }        //属性 pxm
}
```

6.8.6　方法设计

方法是包含一系列的代码块。从本质上来讲，方法就是和类相关联的动作，是类的外部界面。用户可以通过外部界面来操作类的私有字段。

1. 方法的定义

定义方法的基本格式如下：

```
修饰符 返回类型 方法名(参数列表)
{
    //方法的具体实现;
}
```

其中，如果省略“修饰符”，默认为 private。“返回类型”指定该方法返回数据的类型，它可以是任何有效的类型。如果方法不需要返回一个值，其返回类型必须是 void。“参数列表”是用逗号分隔的类型、标识符对。这里的参数是形参，本质上是变量，用来在调用方法时接收实参传给方法的值，如果方法没有参数，那么“参数列表”为空。

2. 方法的返回值

方法可以向调用方返回某一特定的值。如果返回类型不是 void 则该方法可以用 return 关键字来返回值，return 还可以用来停止方法的执行。

3. 方法的参数

方法中的参数是保证不同的方法间互动的重要桥梁，方便用户对数据的操作。C＃中方法的参数有 4 种类型。

(1) 值参数

值参数不含任何修饰符，当利用值向方法传递参数时，编译程序给实参的值做一份备份，并且将此备份传递给该方法对应的形参，被调用的方法不会修改内存中实参的值，所以使用值参数时是可以保证实参的安全性的。

例如，前面 Class1 类中 setdata 方法中的参数就是值参数。

(2) 引用型参数

以 ref 修饰符声明的参数属引用型参数。引用型参数本身并不创建新的存储空间，而是将实参的存储地址传递给形参。所以对形参的修改会影响原来的实参。在调用方法前，引用型实参必须被初始化，同时在调用方法时，对应引用型参数的实参也必须使用 ref 修饰。

例如，以下定义的 MyClass 类中的 addnum 方法使用了一个引用型参数 num2：

```
public class MyClass
{   int num = 0;
```

```
    public void addnum(int num1,ref int num2)
    {   num2 = num  +  num1; }
}
```

以下语句调用 addnum 方法时实参 x 发生改变：

```
int x = 0;                          //引用实参要置初值
MyClass s  =  new MyClass();
s.addnum(5, ref x);                 //x 的值变为 5
```

(3) 输出参数

以 out 修饰符声明的参数属输出参数。与引用型参数类似，输出型参数也不开辟新的内存区域。与引用型参数的差别在于：调用方法前无须对实参进行初始化。输出型参数用于传递方法返回的数据，out 修饰符后应跟随与形参的类型相同的类型，声明在方法返回后传递的变量被认为经过了初始化。

例如，以下定义的 MyClass 类中的 addnum 方法使用了一个输出参数 num2：

```
public class MyClass
{   int num = 0;
    public void addnum(int num1,out int num2)
    {   num2 = num  +  num1; }
}
```

以下语句调用 addnum 方法时实参 x 发生改变：

```
int x;                              //输出型实参不必置初值
MyClass s  =  new MyClass();
s.addnum(5, out x);                 //x 的值变为 5
```

(4) 数组型参数

以 params 修饰符声明数组型参数。params 关键字可以指定在参数数目可变处采用参数的方法参数。在方法声明中的 params 关键字之后不允许任何其他参数，并且在方法声明中只允许一个 params 关键字。数组型参数不能再有 ref 和 out 修饰符。

例如，以下定义的 MyClass 类中的 addnum 方法使用了一个数组型参数 b：

```
public class MyClass
{   int num = 10;
    public void addnum(ref int sum,params int[] b)
    {   sum  =  num;
        foreach (int item in b)
            sum  +=  item;
    }
}
```

以下语句求实参数组 a 的所有元素之和：

```
int x  =  0;
MyClass s  =  new MyClass();
s.addnum(ref x,1,2,3);              //x 的值为 6
s.addnum(ref x,1,2);                //x 的值为 3
s.addnum(ref x,1);                  //x 的值为 1
```

4. 可选参数

所谓可选参数是指在调用方法时可以包含这个参数，也可以忽略它。为了表明每个参数是可选的，需要在方法定义时为它提供参数默认值。指定默认值的语法和初始化本地变量的语法一样。例如，声明如下类：

```
class MyClass
{   public int add(int a, int b = 1, int c = 2)
    {  return a + b + c; }
}
```

该类中 add 方法有两个可选参数，有如下代码：

```
MyClass s = new MyClass();
int x = s.add(5);
int y = s.add(5, 6);
int z = s.add(5, 6, 7);
```

计算过程是 x=5+1+2=8，y=5+6+2=13，z=5+6+7=18。从中看到，当可选参数没有给出实参时，自动取可选参数的默认值。

不是所有的参数类型都可以作为默认值，其规定如下：

- 只有值类型的默认值在编译的时候可以确定，就可以使用值类型作为可选参数；
- 只有在默认值是 null 时，引用类型才可以作为可选参数来使用。

如果一个方法包含必填参数、可选参数和 params 参数，则必填参数必须在可选参数之前声明，而 params 参数必须在可选参数之后声明。其一般格式如下：

```
un(int x,double y, …,int op1 = 1,double op2 = 2.5, …,params int [] arr)
     必填参数                可选参数                params参数
```

5. 方法的重载

方法的重载是指调用同一方法名，但是使用不同的数据类型的参数或参数的次序不一致。只要一个类中有两个以上的同名方法，且使用的参数类型或个数不同，编译器就可以判断在哪种情况下调用哪种方法了。

为此，C#中引入了成员签名的概念。成员签名包含成员的名称和参数列表，每个成员签名在类型中必须是唯一的，只要成员的参数列表不同，成员的名称可以相同。如果同一个类有两个或多个这样的成员（方法、属性、构造函数等），它们具有相同的名称和不同的参数列表，则称该同类成员进行了重载，但它们的成员签名是不同的。

例如，下面的代码实现了 MethodTest 方法的重载（假设都是某个类的成员），它们是不同的成员签名：

```
public int MethodTest(int i,int j)      //重载方法 1
{
    //代码
}
public int MethodTest(int i)            //重载方法 2
{
    //代码
}
public string MethodTest(string str)    //重载方法 3
```

```
{
    //代码
}
```

6.8.7　委托和事件

1. 委托简介

委托是一种安全的封装方法的类型，类似于 C/C++ 中的函数指针，通过委托可以将方法作为参数或变量使用，从而可以采用统一的格式调用多个方法。与 C/C++ 中的函数指针不同，委托是类型安全的。

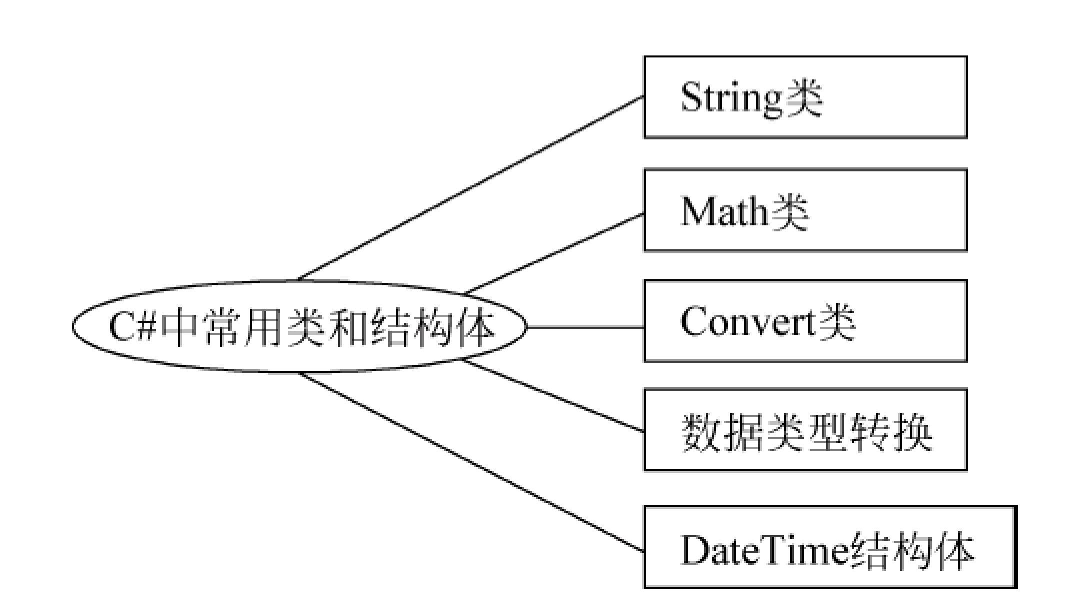
2. 事件简介

事件是一种用于类和类之间传递消息或触发新的行为的编程方式。事件可以看成类的委托，能够把控件和可执行代码联系在一起，如用户单击 Button 控件触发 Click 事件后就执行相应的事件处理代码。

事件的声明通过委托来实现。先定义委托，再用委托定义事件，触发事件的过程实质是调用委托。

6.9　C#中常用类和结构体

知识梳理



6.9.1　String 类

前面介绍过，String 类型表示字符串，实际上，string 是 .NET Framework 中的 String 类的别名。String 类型定义了相等运算符（== 和 !=）用于比较两个 string 对象，另外，"+"运算符用于连接字符串，"[]"运算符可以用来访问 String 中的各个字符。

String 类位于 System 命名空间中，用于字符串的处理。String 类常用的属性如表 6.9 所示，常用的方法如表 6.10 所示，使用这些属性和方法会为字符串的处理带来极大的方便。

表 6.9　String 的常用属性及其说明

属　性	说　明
Chars	获取此字符串中位于指定字符位置的字符
Length	获取此字符串中的字符数

表 6.10　String 的常用方法及其说明

方法	方法类型	说　　明
Compare	静态方法	比较两个指定的 String 对象
Concat		连接 String 的一个或多个字符串
Format		将指定的 String 中的每个格式项替换为相应对象的值的文本等效项
CompareTo	非静态方法	非静态方法。将此字符串与指定的对象或 String 进行比较，并返回两者相对值的指示
Contains		返回一个值，该值指示指定的 String 对象是否出现在此字符串中
Equals		确定两个 String 对象是否具有相同的值
IndexOf		返回 String 或一个或多个字符在此字符串中的第一个匹配项的索引
Insert		在该 String 中的指定索引位置插入一个指定的 String
Remove		从该 String 中删除指定个数的字符
Replace		将该 String 中的指定 String 的所有匹配项替换为其他指定的 String
Split		返回包含该 String 中的子字符串（由指定 Char 或 String 数组的元素分隔）的 String 数组
Substring		从此字符串中检索子字符串
ToLower		返回该 String 转换为小写形式的副本
ToUpper		返回该 String 转换为大写形式的副本
Trim		从此字符串的开始位置和末尾移除一组指定字符的所有匹配项

注意：一个类的方法有静态方法和非静态方法之分。对于静态方法，只能通过类名来调用，对于非静态方法，需通过类的对象来调用。

6.9.2　Math 类

Math 类位于 System 命名空间中，包含了实现 C＃中常用算术运算功能的方法。这些方法都是静态方法，可通过“Math.方法名(参数)”来使用，其中常用的方法如表 6.11 所示。

表 6.11　Math 类的常用方法

方　　法	说　　明
Abs	返回指定数字的绝对值
Acos	返回余弦值为指定数字的角度
Asin	返回正弦值为指定数字的角度
Atan	返回正切值为指定数字的角度
Atan2	返回正切值为两个指定数字的商的角度
Ceiling	返回大于或等于指定数字的最小整数
Cos	返回指定角度的余弦值
Cosh	返回指定角度的双曲余弦值
DivRem	计算两个数字的商，并在输出参数中返回余数
Exp	返回 e 的指定次幂
Floor	返回小于或等于指定数字的最大整数
Log	返回指定数字的对数
Log10	返回指定数字以 10 为底的对数
Max	返回两个指定数字中较大的一个
Min	返回两个数字中较小的一个

续表

方　法	说　明
Pow	返回指定数字的指定次幂
Round	将值舍入到最接近的整数或指定的小数位数
Sign	返回表示数字符号的值
Sin	返回指定角度的正弦值
Sinh	返回指定角度的双曲正弦值
Sqrt	返回指定数字的平方根
Tan	返回指定角度的正切值
Tanh	返回指定角度的双曲正切值
Truncate	计算一个数字的整数部分

6.9.3 Convert 类

Convert 类位于 System 命名空间中，用于将一个值类型转换成另一个值类型。这些方法都是静态方法，可通过“Convert.方法名(参数)”来使用，其中常用的方法如表 6.12 所示。

表 6.12 Convert 类的常用方法

方　法	说　明
ToBoolean	将数据转换成 Boolean 类型
ToDataTime	将数据转换成日期时间类型
ToInt16	将数据转换成 16 位整数类型
ToInt32	将数据转换成 32 位整数类型
ToInt64	将数据转换成 64 位整数类型
ToDouble	将数据转换成 Double 类型
ToObject	将数据转换成 Object 类型
ToString	将数据转换成 String 类型

6.9.4 数据类型转换

1. 数值类型与字符串之间转换

将数字字符串转换为数值类型使用数值类型的 Parse 方法。例如：

```
int n;
float f;
double d;
string s1 = "123",s2 = "1.2",s3 = "125.8";
s1 = int.Parse(n);              //将字符串 s1 转换 int 类型
s2 = float.Parse(f);            //将字符串 s2 转换 float 类型
s3 = double.parse(d);           //将字符串 s3 转换 double 类型
```

数值类型转换为字符串使用 ToString 方法。例如：

```
float f = 1.25;
double = 100.6;
string s1,s2,s3;
s1 = n.ToString();              //将 int 类型转换为字符串 s1
```

```
s2 = f.ToString();                  //将 float 类型转换为字符串 s2
s3 = d.ToString();                  //将 double 类型转换为字符串 s3
```

2. 使用 Convert 方法进行转换

例如：

```
double d = 12.5,d1;
int n = 8,n1;
string s = "68",s1,s2;
n1 = Convert.ToInt32(s);            //n1 = 68
d1 = Convert.ToDouble(s);           //d1 = 68.0
s1 = Convert.ToString(n);           //s1 = "8"
s2 = Convert.ToString(d);           //s2 = "12.5"
```

注意：使用 Convert 方法将字符串转换为整数时，字符串中一定是整数字符串。例如，string s=Convert.ToInt("68.2")是错误的，因为"68.2"不是整数字符串。

6.9.5　DateTime 结构体

DateTime 结构体位于 System 命名空间中，DateTime 值类型表示值范围在公元 0001 年 1 月 1 日午夜 12:00:00 到公元 9999 年 12 月 31 日晚上 11:59:59 之间的日期和时间。可以通过以下语法格式定义一个日期时间变量：

```
DateTime 日期时间变量 = new DateTime(年,月,日,时,分,秒);
```

例如，以下语句定义了两个日期时间变量：

```
DateTime d1 = new DateTime(2009,10,1);
DateTime d2 = new DateTime(2009,10,1,8,15,20);
```

其中，d1 的值为 2009 年 10 月 1 日零点零分零秒，d2 的值为 2009 年 10 月 1 日 8 点 15 分 20 秒。

DateTime 结构体的常用属性如表 6.13 所示，常用方法如表 6.14 所示。

表 6.13　DateTime 结构体的常用属性

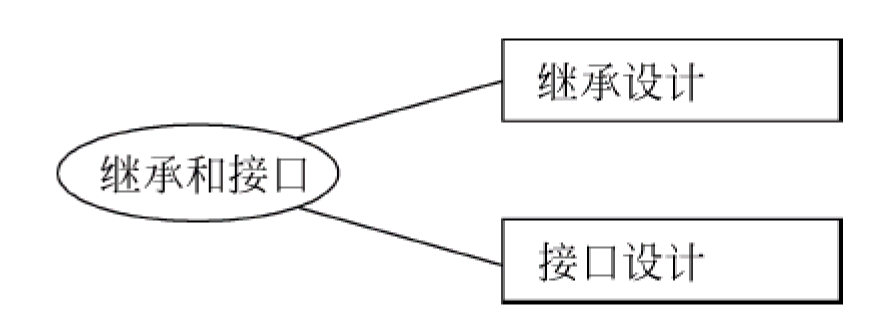
属　性	说　明
Date	获取此实例的日期部分
Day	获取此实例所表示的日期为该月中的第几天
DayOfWeek	获取此实例所表示的日期是星期几
DayOfYear	获取此实例所表示的日期是该年中的第几天
Hour	获取此实例所表示日期的小时部分
Millisecond	获取此实例所表示日期的毫秒部分
Minute	获取此实例所表示日期的分钟部分
Month	获取此实例所表示日期的月份部分
Now	获取一个 DateTime 对象，该对象设置为此计算机上的当前日期和时间，表示为本地时间
Second	获取此实例所表示日期的秒部分
TimeOfDay	获取此实例的当天的时间
Today	获取当前日期
Year	获取此实例所表示日期的年份部分

表 6.14　DateTime 结构体的常用方法

方　法	方法类型	说　明
Compare	静态方法	比较 DateTime 的两个实例,并返回它们相对值的指示
DaysInMonth		返回指定年和月中的天数
IsLeapYear		返回指定的年份是否为闰年的指示
Parse		将日期和时间的指定字符串表示转换成其等效的 DateTime
AddDays	非静态方法	将指定的天数加到此实例的值上
AddHours		将指定的小时数加到此实例的值上
AddMilliseconds		将指定的毫秒数加到此实例的值上
AddMinutes		将指定的分钟数加到此实例的值上
AddMonths		将指定的月份数加到此实例的值上
AddSeconds		将指定的秒数加到此实例的值上
AddYears		将指定的年份数加到此实例的值上
CompareTo		将此实例与指定的对象或值类型进行比较,并返回两者相对值的指示

6.10　继承和接口

知识梳理



6.10.1　继承设计

1. 什么是继承

为了对现实世界中的层次结构进行模型化,于是面向对象的程序设计技术引入了继承的概念。继承是面向对象程序设计最重要的特征之一。任何类都可以从另外一个类来继承而来,即这个类拥有它所继承类的所有成员。C#提供了类的继承机制,但C#只支持单继承不支持多重继承,即在C#中一次只允许继承一个类,不允许继承多个类。

一个类从另一个类派生出来时,称为派生类或子类,被派生的类称为基类或父类。派生类从基类那里继承特性,派生类也可以作为其他类的基类,从一个基类派生出来的多层类形成了类的层次结构。

与C++不同,C#中仅允许单个继承,也就是说,类只能从一个基类继承实现。C#中的继承具有以下特点:

- C#中只允许单继承,即一个派生类只能有一个基类;
- C#中继承是可以传递的,如果C从B派生,B从A派生,那么C不仅继承B的成员,还继承A的成员;
- C#中派生类可以添加新成员,但不能删除基类的成员;
- C#中派生类不能继承基类的构造函数和析构函数,但能继承基类的属性;

- C#中派生类可以隐藏基类的同名成员，如果在派生类可以隐藏了基类的同名成员，基类该成员在派生类中就不能被直接访问，只能通过“base.基类方法名”来访问；
- C#中派生类对象也是基类的对象，但基类对象却不一定是基派生类的对象。也就是说，基类的引用变量可以引用基派生类对象，而派生类的引用变量不可以引用基类对象。

2. 派生类的声明

派生类的声明格式如下：

```
[类修饰符] class 派生类: 基类;
```

C#中派生类可以从它的基类中继承字段、属性、方法和事件等，实际上除了构造函数和析构函数，派生类隐式地继承了基类的所有成员。

下面来看一个例子。先声明一个基类：

```
class A
{   private int n;                  //私有字段
    protected int m;                //保护的字段
    public void afun()              //公有方法
    {
        //方法的代码
    }
}
```

再声明一个 B 类继承 A 类，注意继承是用“:”来表示的：

```
class B : A
{   private int x;                  //私有字段
    public void bfun()              //公有方法
    {
        //方法的代码
    }
}
```

A、B 类的继承关系如图 6.14 所示。有以下代码：

```
B b = new B();          //定义对象并实例化
b.afun();
```

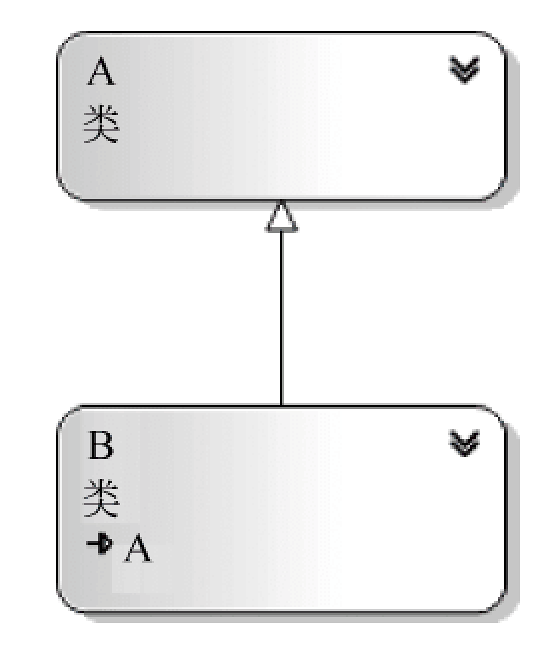


图 6.14　A、B 两个类的继承关系

从中可以看出 A 类的 afun()方法在 B 类中不用重新编写。因为 B 类继承了 A 类，所以可以通过类 B 类的对象调用它。

3. 基类成员的可访问性

派生类将获取基类的所有非私有数据和行为以及新类为自己定义的所有其他数据或行为。在前面的例子中，基类 A 中保护的字段 m 和公有方法 afun 都被继承到派生类 B 类中，这样在 B 类中隐含有保护的字段 m 和公有方法 afun。但基类 A 中的私有字段 n 不能被继承到派生类 B 中。

所以，如果希望在派生类中隐藏某些基类的成员，可以在基类中将这些成员设为 private 访问成员。

4. 使用 sealed 修饰符来禁止继承

C＃中提供了 sealed 关键字用来禁止继承。要禁止继承一个类，只需要在声明类时加上 sealed 关键字就可以了，这样的类称为密封类。例如：

```
sealed class 类名
{
     ⋮
}
```

这样就不能从该类派生任何子类。

5. 多态性

多态性也是面向对象程序设计最重要的特性之一。多态性是指发出同样的消息(如方法调用)被不同类型的对象接收时可能导致不同的行为，运算符重载和方法重载都属于多态性的表现形式。本节介绍采用虚方法实现多态性，也就是子类继承父类，并重写父类的方法，从而实现了不同的操作。

(1) 隐藏基类方法

C＃中可以为每个类的每个方法给出特定的代码，而且还需要让程序能够调用正确的方法。当派生类从基类继承时，它会获得基类的所有方法、字段、属性和事件。若要更改基类的数据和行为，有两种选择，可以使用新的派生成员替换基成员或可以重写虚拟的基成员。本小节介绍前一种方法，在下一小节介绍后一种方法。

在使用新的派生方法替换基方法时应使用 new 关键字。例如：

```
class A                              //声明基类 A
{   public void fun()
    {     //输出 A;   }
}
class B:A                            //从 A 类派生 B 类
{   new public void fun()            //隐藏基类方法 fun
    {     //输出 B   }
}
```

执行以下语句：

```
B b = new B();
b.fun();                             //结果输出 B
```

从结果中看到，b.fun()语句调用的是类 B 的方法。如果类 B 中 fun 方法定义没有使用 new 关键字，编译时会给出警告信息：“B.fun()隐藏了继承的成员 A.fun()，如果是有意隐藏，请使用关键字 new”。

如果要在派生类中使用基类中隐藏了的成员，可以使用“base.成员名”。

(2) 重写基类方法

重写是指在子类中编写有相同名称和参数的方法，或者说重写是在子类中对父类的方法进行修改或重新编写。重写和重载是不同的，重载是指编写(在同一个类中)具有相同的名称、却有不同的参数即有不同签名的方法。也就是说，重写是指子类中的方法与基类中的方法具有相同的签名，而重载方法具有不同的签名。

重写父类方法的过程如下：

- 在父类中使用 virtual 关键字把某个方法定义为虚方法；
- 在子类中使用 override 关键字重写父类的虚方法。

【练一练】 在 CH6 网站中设计一个网页 webform4.aspx，说明构造函数的使用方法。

其设计步骤如下：

① 打开 CH6 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH6”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform4.aspx，其他保持默认项，单击“添加”按钮，创建一个空的网页。

② 打开 Class1.css 类文件，添加如下 4 个类的代码：

```
public class Rectangle                                //长方形类
{   protected double x, y;
    public Rectangle() { }
    public Rectangle(double x1, double y1)
    {  x = x1; y = y1; }
    public virtual double Area()                      //求面积方法
    {  return x * y; }
}
public class Circle : Rectangle                       //从 Rectangle 类派生圆类
{   public Circle(double r) : base(r, 0) { }
    public override double Area()                     //求面积方法
    {  return Math.PI * x * x; }
}
public class Sphere : Rectangle                       //从 Rectangle 类派生圆球体类
{   public Sphere(double r) : base(r, 0) { }
    public override double Area()                     //求面积方法
    {  return 4 * Math.PI * x * x; }
}
public class Cylinder : Rectangle                     //从 Rectangle 类派生圆柱体类
{   public Cylinder(double r, double h) : base(r, h) { }
    public override double Area()                     //求面积方法
    {  return 2 * Math.PI * x * x + 2 * Math.PI * x * y; }
}
```

其中，Rectangle 是基类，该基类包含 x(长)和 y(宽)两个保护的字段和计算图形面积或表面积的 Area()虚方法，从它派生出 Circle、Cylinder 和 Sphere 类。每个派生类都有各自的 Area()重写实现。根据与此方法关联的对象，通过调用正确的 Area()实现，该程序为每个图形计算并显示正确的面积或表面积。

③ 打开 webform4.aspx 网页，设计其界面如图 6.15 所示，其中主要包含 6 个文本框(从上到下、从左到右 ID 分别为 TextBox1～TextBox6)、1 个命令按钮 Button1 和 1 个标签 Label1。

④ 网页上设计如下事件处理方法：

```
protected void Button1_Click(object sender, EventArgs e)
{   double x = double.Parse(TextBox1.Text.Trim());
    double y = double.Parse(TextBox2.Text.Trim());
    double r1 = double.Parse(TextBox3.Text.Trim());
    double r2 = double.Parse(TextBox4.Text.Trim());
    double r3 = double.Parse(TextBox5.Text.Trim());
```

```
        double h = double.Parse(TextBox6.Text.Trim());
        Rectangle t = new Rectangle(x, y);
        Rectangle c = new Circle(r1);
        Rectangle s = new Sphere(r2);
        Rectangle l = new Cylinder(r3, h);
        string mystr = "求解结果如下：<br>";
        mystr += "长方形面积=" + Math.Round(t.Area(),2) + ",";
        mystr += "圆面积=" + Math.Round(c.Area(),2) + "<br>";
        mystr += "圆球体表面积=" + Math.Round(s.Area(),2) + ",";
        mystr += "圆柱体表面积=" + Math.Round(l.Area(),2);
        Label1.Text = mystr;
    }
```

⑤ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，输入各个值，单击“求面积”命令按钮，其执行结果如图 6.16 所示。



图 6.15 webform4 网页的设计界面



图 6.16 webform4 网页的执行界面

6.10.2 接口设计

C# 不像 C++ 那样支持类的多继承，为此提供了接口可以实现 C++ 中多继承的功能。

1. 接口设计概述

C# 中接口是类之间交互内容的一个抽象，把类之间需要交互的内容抽象出来定义成接口，可以更好的控制类之间的逻辑交互。

接口只包含方法、委托或事件的签名，方法的实现是在实现接口的类中完成的。例如，如下代码声明了一个接口：

```
interface Ia                                //声明接口 Ia
{
    float getarea();                        //接口成员声明
}
```

其中 getarea 方法只有声明部分。设计一个 Rectangle 类实现该接口：

```
public class Rectangle : Ia                 //类 A 继承接口 Ia
```

```
{    float x,y;
     public Rectangle(float x1, float y1)              //构造函数
     {  x = x1; y = y1;  }
     public float getarea()                            //隐式接口成员实现,必须使用 public
     {  return x * y;  }
}
```

这样,就可以定义 Rectangle 类的对象并调用 getarea 方法了:

```
Rectangle box1 = new Rectangle(2.5, 3.0);          //定义一个类实例
TextBox1.Text = box1.getarea();                    //文本框中显示长方形面积为 7.5
```

一个接口可以从零个或多个接口中继承。一个接口可以被多个类继承,在这些类中实现该接口的成员,这样接口就起到提供统一界面的作用。

2. 接口设计应用

前面介绍过 C#数组和集合,集合是 ArralList 类的对象。集合排序是十分有用的,对于简单类型的集合元素,可以直接使用 ArrayList 类的 Sort 方法排序,而对于复杂类型的集合元素,需要使用 C#的 Comparable 等系统接口。

IComparable 接口定义通用的比较方法,由值类型或类实现以创建类型特定的比较方法。其公共成员有 CompareTo,它用于比较当前实例与同一类型的另一对象。其使用语法格式如下:

```
int CompareTo(Object obj)
```

其中,obj 表示与此实例进行比较的对象。其返回值是一个 32 位有符号整数,指示要比较的对象的相对顺序。返回值的含义如下:

- 小于零:此实例小于 obj。
- 零:此实例等于 obj。
- 大于零:此实例大于 obj。

IComparable 接口的 CompareTo 方法提供默认排序次序,如果需要改变其排序方式,可以在相关类中实现 CompareTo 方法,以定制其比较功能。

【练一练】 在 CH6 网站中设计一个网页 webform5.aspx,说明使用 Comparable 接口实现集合排序的方法。

其设计步骤如下:

① 打开 CH6 网站,选择“网站|添加新项”菜单命令,出现“添加新项-CH6”对话框,在中间列表中选择“Web 窗体”,将文件名称改为 webform5.aspx,其他保持默认项,单击“添加”按钮,创建一个空的网页。

② 打开 Class1.css 类文件,添加如下代码:

```
public class student : IComparable                     //student 类从接口派生
{    public int xh { get;set; }                        //学号属性
     public string xm { get; set; }                    //姓名属性
     public int fs { get;set; }                        //分数属性
     public student() { }
     public student(int no, string name, int degree)   //构造函数
     {
         xh = no; xm = name; fs = degree;
```

```
        }
        public int CompareTo(object obj)                    //实现接口方法
        {   student s = (student)obj;                        //转换为 student 实例
            if (fs < s.fs) return 1;
            else if (fs == s.fs) return 0;
            else return -1;
        }
    }
```

③ 在本网页中拖曳一个标签 Label1 和一个命令按钮 Button1，设置好字体和颜色属性。设置如下处理方法：

```
protected void Button1_Click(object sender, EventArgs e)
{   int i;
    ArrayList myarr = new ArrayList();
    student st = new student(1,"王华",88);
    myarr.Add(st);
    st = new student(2,"李明",78);
    myarr.Add(st);
    st = new student(3, "章冰", 92);
    myarr.Add(st);
    myarr.Sort();
    Label1.Text = "";
    foreach (student item in myarr)
        Label1.Text += item.xh + " " + item.xm + " "
         + item.fs.ToString() + "<br>";
}
```

④ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，单击“按分数递减排序”命令按钮，其执行结果如图 6.17 所示。

由于 myarr 集合（ArrayList 类对象）中存放的是 student 类对象，调用 myarr.Sort()方法时不知道按哪个字段排序，所以需要在 student 类中实现 IComparable 接口的 CompareTo 方法，这样，在调用 myarr.Sort()方法时，自动采用 CompareTo 方法进行数组元素的大小比较。



图 6.17　webform5 网页的执行界面

6.11　练　习　题

1. 单项选择题

(1) C#的数据类型有(　　)。

A. 值类型和调用类型　　　　B. 值类型和引用类型

C. 引用类型和关系类型　　　D. 关系类型和调用类型

(2) 以下(　　)语句是创建数组的正确语句。

A. floatf[]=newfloat[6];　　　B. floatf[6]=newfloat[6];

C. float[]f=newfloat[6];　　　D. floatf=newfloat[6];

(3) 在 C#中,关于 while 和 do-while,以下说法正确的是(　　)。

A. while 语句先执行然后判断条件是否成立

B. while 语句最少的循环次数是一次

C. do-while 语句先执行然后判断条件是否成立

D. do-while 语句最少的循环次数是 0 次

(4) 在 C#中 using 关键字的作用是(　　)。

A. 定义命名空间　B. 新建实例　C. 调用类　D. 引入命名空间

(5) 在类的定义中,类的(　　)描述了该类的对象的行为特征。

A. 方法　B. 类名　C. 命名空间　D. 私有字段

(6) 在 C#中 new 关键字的作用是(　　)。

A. 新建对象实例　B. 定义命名空间　C. 调用类　D. 引入命名空间

(7) 调用重载方法时,系统根据(　　)来选择具体的方法。

A. 方法名　B. 参数的个数和类型

C. 参数名及参数个数　D. 方法的返回值类型

(8) 以下几个函数,(　　)是重载函数。

① voidf1(int)　② intf1(int)　③ intf1(int,int)　④ floatk(int)

A. 4 个全　B. ①和④　C. ②和③　D. ③和④

(9) 下列关于构造函数的描述正确的是(　　)。

A. 构造函数可以声明返回类型　B. 构造函数不可以用 private 修饰

C. 构造函数必须与类名相同　D. 构造函数不能带参数

(10) 以下关于 ref 和 out 的叙述中错误的是(　　)。

A. 使用 ref 参数,传递到 ref 形数的实参必须最先初始化

B. 使用 out 参数,传递到 out 形数的实参必须最先初始化

C. 使用 ref 参数,必须将实参作为 ref 参数显式传递到方法

D. 使用 out 参数,必须将实参作为 out 参数显式传递到方法

(11) 有如下类声明:

```
public class People
{   int age = 8;
    public int Age
    {
        get { return age; }
    }
}
```

执行以下语句后 TextBox1 中的结果是(　　)。

```
People p = new People();
p.Age++;
TextBox1.Text = p.Age.ToString();
```

A. 8　B. 9　C. 程序有编译错误　D. 0

(12) 类的字段和方法的默认访问修饰符是(　　)。

A. public　B. private　C. protected　D. internal

(13) 假设类 B 继承了类 A,下列说法错误的是(　　)。

A. 类 B 中的成员可以访问类 A 中的公有成员

B. 类 B 中的成员可以访问类 A 中的保护成员

C. 类 B 中的成员可以访问类 A 中的私有成员

D. 类 B 中的成员可以访问类 A 中的静态成员

(14) 以下接口和类的叙述中正确的是________。

A. 类可以继承而接口不能　　B. 类不能继承而接口可以

C. 类可以多继承而接口不能　　D. 类不能多继承而接口可以

(15) 在 C# 程序中可以使用 try-catch 机制来处理程序出现的(　　)错误。

A. 语法　　B. 执行　　C. 逻辑　　D. 拼写

2. 问答题

(1) 什么是装箱和拆箱?

(2) 在 C# 中,String str = null 与 String str = ""有什么区别?

(3) 简述类字段和方法前加 static 的作用。

(4) 简述 private、protected、public、internal 类成员修饰符的访问权限。

(5) try {}里有一个 return 语句,那么紧跟在这个 try 后的 finally {}里的代码会不会被执行,什么时候被执行,在 return 前还是后?

(6) 简述 override 与重载的区别。

(7) C# 中的委托是什么? 事件是不是一种委托?

(8) 在命令按钮 Button1 上设计一个单击事件处理方法,在标签 Label1 中显示 1~100 之间的所有素数。

(9) 在命令按钮 Button1 上设计一个单击事件处理方法,在标签 Label1 中显示 10 个 1~100 之间的不重复的随机数(使用 Random 类的 Next 方法产生随机数)。

(10) 一个网页设计有如下类:

```
class A
{   public int a { get;set;}
    public A(int x)
    {  a = x;  }
}
class B : A
{   public int b { get;set;}
    public B(int x,int y):base(x)
    {  b = 2; }
}
```

网页中 Button1 的单击事件处理方法如下:

```
protected void Button1_Click(object sender, EventArgs e)
{   B obj = new B(1, 2);
    Label1.Text = "a=" + obj.a.ToString() + ",b=" + obj.b.ToString();
}
```

回答如下问题:

① 类 A 中的成员 a 是什么成员?

② 在执行该网页时,用户单击 Button1 命令按钮,在 Label1 中显示的结果是说明?

6.12　上机实验题

在 CH6 网站添加一个 Exp 网页，设计界面如图 6.18 所示，用户单击“产生随机数并排序”命令按钮时，产生 20 个 1～20 的随机数，并递增排序，如图 6.19 所示。

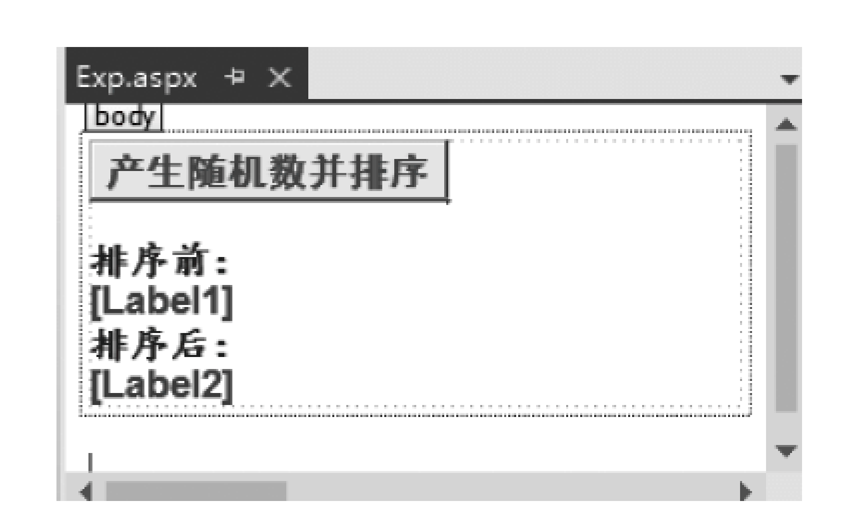


图 6.18　上机实验题网页的设计界面



图 6.19　上机实验题网页的执行界面

第 7 章　ASP .NET 控件

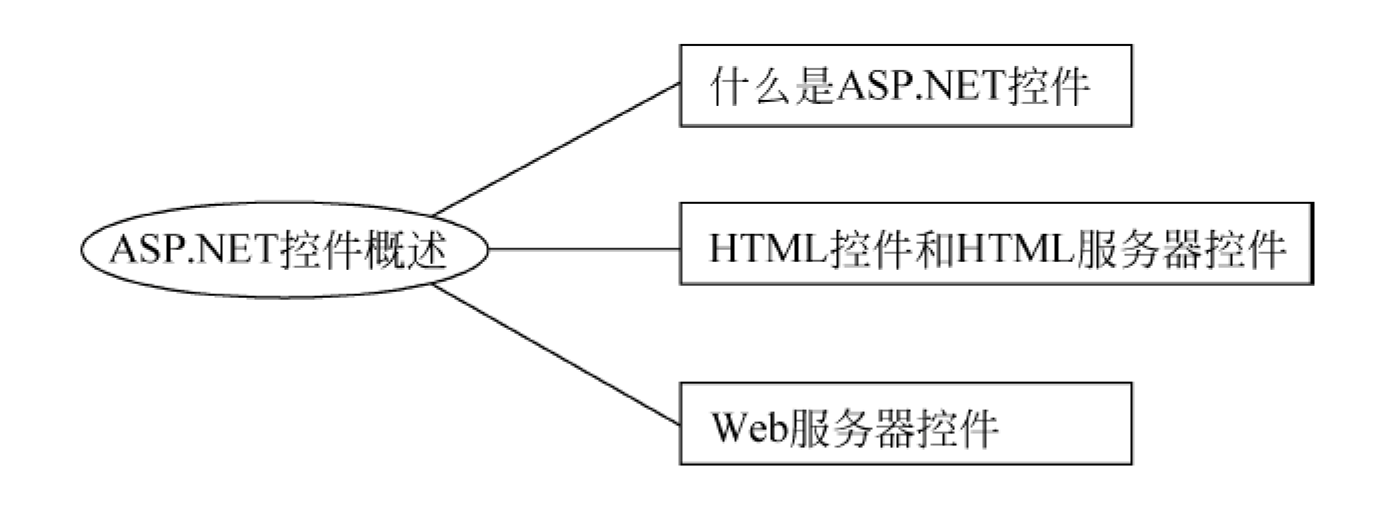


本章指南



7.1　ASP .NET 控件概述

知识梳理



7.1.1　什么是 ASP .NET 控件

使用 ASP .NET 开发网页之所以方便和快捷，关键是它有一组强大的控件库，该控件库中的控件就是 ASP .NET 控件，每个控件都是一个可重用的组件或对象，有自己的属性、方法和可以响应的事件。

在 ASP .NET 网页设计时，工具箱中的所有控件都是 ASP .NET 控件，都可以拖曳到网页中构成网页元素。

ASP .NET 控件分为 Web 服务器控件、HTML 服务器控件和 HTML 控件等类型。

7.1.2　HTML 控件和 HTML 服务器控件

1. HTML 控件

HTML 控件就是通常说的 HTML 标记或静态元素，在 ASP .NET 工具箱的 HTML 类别中列出这类控件，如图 7.1 所示。在前面的网页例子中使用过它们。HTML 控件不能在服务器端控制和访问，只能在客户端通过 JavasSript 等脚本代码来控制和访问。

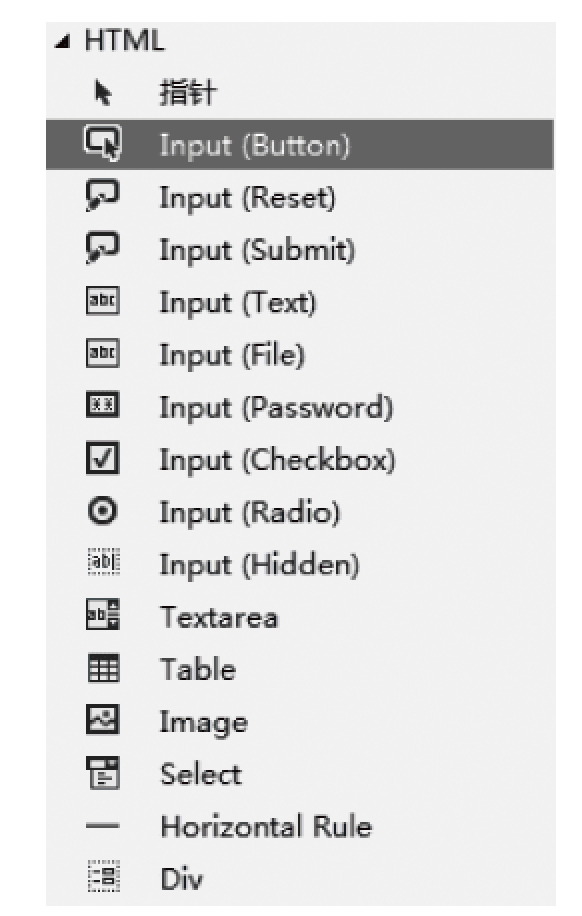


图 7.1　工具箱的 HTML 类别

例如，从工具箱的 HTML 类别向网页中拖曳一个 input (Button)控件，网页中对应的代码如下：

```
<input id = "Button1" type = "button" value = "button" />
```

在执行该网页时，对应的纯 HTML 代码中包含完全相同的上述代码。

2. HTML 服务器控件

ASP .NET 引入了服务器控件的概念，允许开发人员在服务器端访问这些控件，并对它们进行控制。服务器控件就是网页中能够被服务器端代码访问和控制的任何控件，如工具箱的标准部分的控件都是服务器控件，它们都具有 runat="server"属性，id 属性是服务器端代码访问控制的唯一标识号。

ASP .NET 服务器控件都是网页上的对象，采用事件驱动的编程模型，服务器控件的事件处理发生在服务器端而不是客户端，事件的处理需要进行客户端和服务器端的往返，因此，在某些情况下会影响性能。

所谓 HTML 服务器控件，就是在 HTML 控件的基础上加上 runat="server"属性设置所构成的控件。

两者的执行方式是不同的，HTML 控件执行在客户端，而 HTML 服务器控件执行在服务器端。

在执行 ASP .NET 网页，会检查网页元素有无 runat 属性，如果没有，那么 HTML 标记就会被视为字符串，并被送到字符串流等待送到客户端，客户端的浏览器会对其进行解释执行；如果 HTML 标记有 runat="server"属性设置，页面对象会将该控件放入控制器，服务器端的代码就能对其进行控制，等到控制执行完毕后再将 HTML 服务器控件的执行结果转换成 HTML 标记，然后当成字符串流发送到客户端进行解释执行。

例如，网页中的如下代码就是将一个 HTML 控件 Button1 转换为一个 HTML 服务器控件 Button1(仅仅添加了 runat="server"属性设置)：

```
<input id = "Button1" type = "button" value = "button" runat = "server" />
```

HTML 控件转换为服务器控件后，控件的事件在服务器中处理，对应的事件名称也会发生变化。例如，命令按钮 Button 包含 onServerClick 属性，而不是常规 HTML 中使用的 onClick 属性。例如：

```
<input id = "Button1" runat = "server" type = "button" value = "单击"
       onServerClick = "Button1_ServerClick" />
```

就是告诉服务器当命令按钮的单击事件发生时，应调用的事件处理方法是“命令按钮 id_ServerClick”。

7.1.3　Web 服务器控件

Web 服务器控件（或 ASP .NET 服务器控件），是网页编程的基本元素，也是 ASP .NET 特有的。它会按照客户端的情况产生一个或多个 HTML 控件，而不是直接描述 HTML 元素。

例如，从工具箱的“标准”类别中拖曳一个 Button 控件到网页，网页中对应的代码如下：

```
<asp:Button ID = "Button1" runat = "server" Text = "Button" />
```

或

```
<asp:Button ID = "Button1" runat = "server" Text = "Button" ></asp Button>
```

在执行网页时，看到纯 HTML 代码中 Button1 对应的代码如下：

```
<input type = "submit" name = "Button1" value = "Button" id = "Button1" />
```

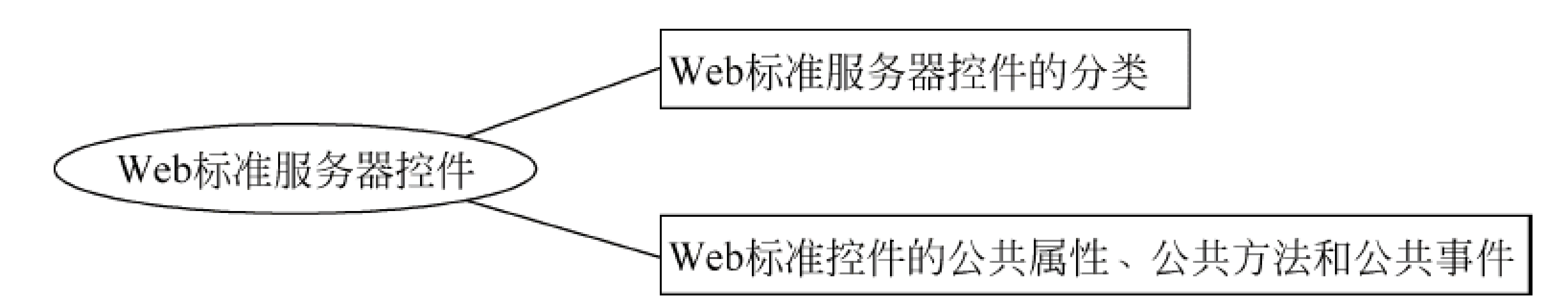
归纳起来，Web 服务器控件和 HTML 服务器控件的区别如下：

- Web 服务器控件提供更加统一的编程接口，如每个 Web 服务器控件都有 Text 属性和 DataBind 方法等；
- 隐藏客户端的不同，使用 Web 服务器控件开发人员可以把更多的精力放在业务上，而不用去考虑客户端的浏览器是 IE 还是 Firefox，或者是移动设备；
- Web 服务器控件可以将状态保存在 ViewState 中，这样页面在从客户端回传到服务器端或从服务器端发送到客户端的过程中都可以保存其控件值等信息；
- 事件处理模型不同，HTML 控件和 HTML 服务器控件的事件处理都是在客户端的页面上，而 Web 服务器控件则是在服务器上。也就是说，HTML 控件和 HTML 服务器控件的事件是由页面来触发的，而 Web 服务器控件则是由页面把表单发回到服务器端，由服务器来处理。

Web 服务器控件又分为 Web 标准服务器控件、验证控件、用户控件和导航控件等。

7.2　Web 标准服务器控件

知识梳理



7.2.1　Web 标准服务器控件的分类

Web 标准服务器控件(简称 Web 标准控件)是开发网页中最常用的控件。位于工具箱的“标准”类别下,共有 29 个。按照功能大致分为 3 类,即 Web 表单控件、列表控件和其他控件。

表单控件放置表单中,构成表单的基本元素。列表控件可以存放多个数据项。

实际上,所有的网页控件都是从 System. Web. UI. Control. WebControls 直接或间接派生而来的,都包含在 System. Web. UI. WebControls 命名空间下。所以,网页程序代码都包含如下语句引用该命名空间:

```
using System.Web.UI.WebControls;
```

7.2.2　Web 标准控件的公共属性、公共方法和公共事件

1. Web 标准控件的公共属性

Web 标准控件的属性可以通过“属性”窗口来设置,也可以通过 HTML 代码实现。Web 标准控件以“asp:”为前缀,ID 属性指定其 ID 值,作为控件的唯一标识,在向客户端呈现时,会将 ID 属性转换为 id 属性。每个 Web 标准服务器控件都有一系列的属性,如表 7.1 所示。

表 7.1　Web 标准控件的常用公共属性

属　　性	说　　明
BackColor	设定对象的背景色,其属性的设定值为颜色名称或是＃RRGGBB 的格式
BorderWidth	设定控件的边框宽度
BorderColor	用来设定边框的颜色
BorderStyle	用来设定控件的边框样式
Enabled	用来决定控件是否正常工作,即是否有效。默认值是 True
Font	用来设置控件的字体及大小
Height、Width	用来设定控件的高和宽,通常单位是 pixel(像素,简写为 px)
TabIndex	用来设定当用户按下 Tab 键时,控件接收焦点的顺序,如果没有设定这个属性,其默认值为零。如果控件的 TabIndex 属性值相同,则是以控件在 ASP .NET 网页中被配置的顺序来决定
ToolTip	设置小提示。在设定本属性后,当用户停留在控件上时就会出现提示的文字
Visible	设置控件是否显示。设定为 False 时,在运行网页时看不到该控件
CssClas	获取或设置由 Web 服务器控件在客户端呈现的级联样式表(CSS)类,即设置控件的样式
SkinID	获取或设置要应用于控件的外观
ViewStateMode	获取或设置此控件的视图状态模式,取值为 Disabled 表示禁用此控件的视图状态,即使父控件已启用了视图状态也是如此;取值为 Enabled 表示启用此控件的视图状态,即使父控件已禁用了视图状态也是如此;取值为 Inherit(默认值)表示从父控件继承 ViewStateMode 的值

2. Web 标准控件的公共方法

Web 标准服务器控件的公共方法如表 7.2 所示。

表 7.2 Web 标准控件的常用公共方法

方　　法	说　　明
ApplyStyleSheetSkin	将网页样式表中定义的样式属性应用到控件
DataBind	将数据源绑定到被调用的服务器控件及其所有子控件
Dispose	使服务器控件得以在从内存中释放之前执行最后的清理操作
FindControl	在当前的命名容器中搜索指定的服务器控件
Focus	为控件设置输入焦点
GetType	获取当前实例的类型
HasControls	确定服务器控件是否包含任何子控件
Render	将控件呈现给指定的 HTML 编写器
RenderControl	生成服务器控件 HTML 输出

3. Web 标准控件的公共事件

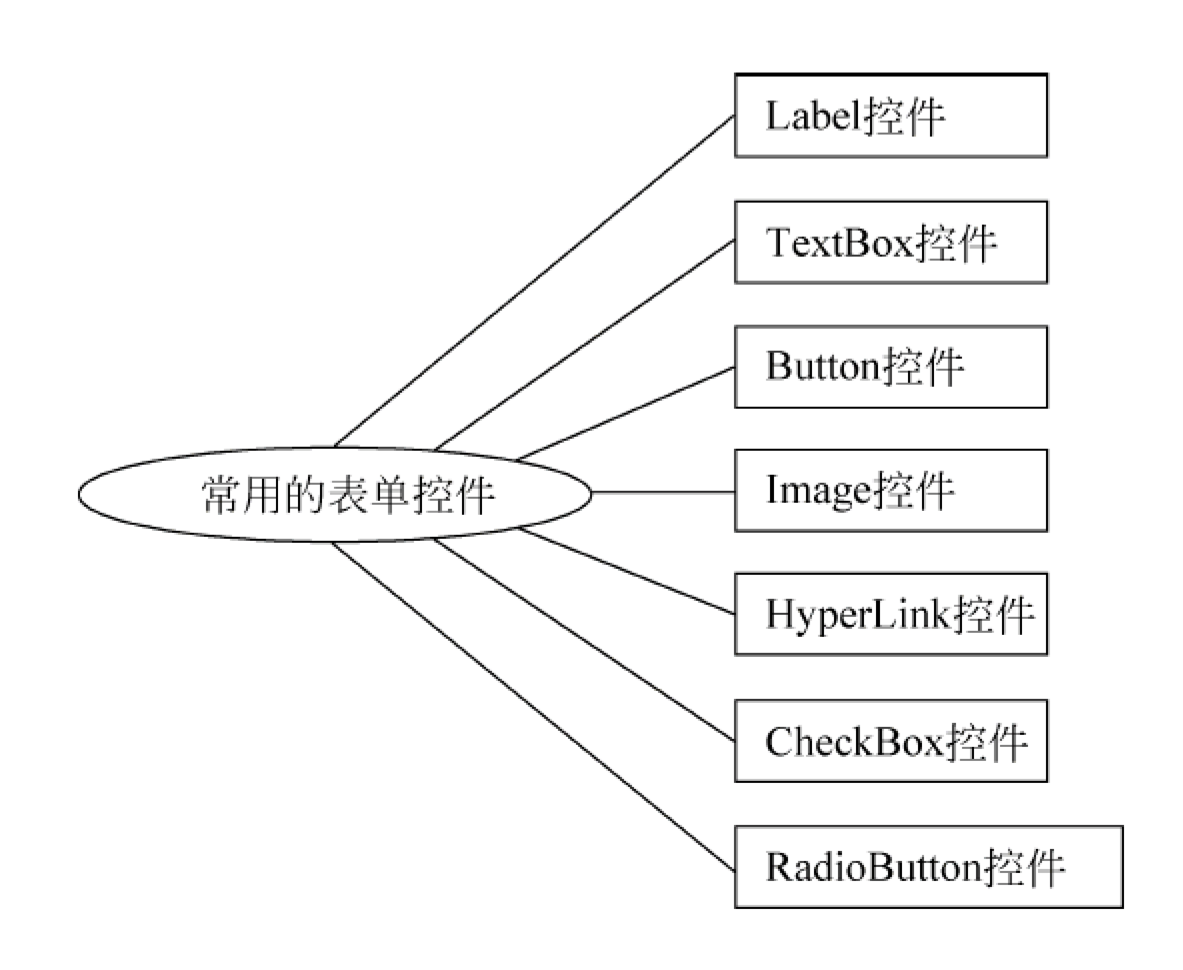
Web 标准服务器控件的公共事件如表 7.3 所示。

表 7.3 Web 标准控件的常用公共事件

事　　件	说　　明
DataBinding	当服务器控件绑定到数据源时引发
Disposed	当从内存释放服务器控件时引发，这是请求 ASP .NET 网页时服务器控件生存期的最后阶段
Init	当服务器控件初始化时引发，初始化是控件生存期的第一步
Load	当服务器控件加载网页时引发
PreRender	在加载控件对象之后、呈现之前引发
Unload	当服务器控件从内存中卸载时引发

7.3 常用的表单控件

知识梳理



7.3.1　Label 控件

Label 控件又称为标签控件。在网页执行中希望显示不能被用户更改的文本，或当触发事件时，某一段由相应事件处理过程更改的文本，则可以使用标签控件。开发人员可以非常方便地将标签控件拖曳到网页。拖曳到网页后，该网页自动生成一段标签控件的声明代码。例如：

```
<asp:Label ID = "Label1" runat = "server" Text = "Label" />
```

上述代码中，声明了一个标签控件，并将这个标签控件的 ID 属性设置为默认值 Label1。由于该控件是服务器端控件，所以在控件属性中包含 runat="server"属性。该代码还将标签控件的文本初始化为“Label”，可以修改该属性以便显示不同文本内容。

标签的 Text 属性包含的字符串可以有多行，可以包含 HTML 标记。例如：

```
<asp:Label ID = "Label1" runat = "server" BorderStyle = "Ridge"
   Text = " 我的标签控件:<br>Label1" Font - Bold = "True"
   Font - Names = "仿宋" Font - Size = "Large" ForeColor = "Fuchsia" />
```

上述标签 Label1 在浏览器中显示的外观如图 7.2 所示。

如果开发人员只是为了显示一般的文本或静态文字，不推荐使用 Label 控件，因为网页的服务器控件过多，会导致性能问题。使用静态的 HTML 文本能够让网页速度更快。



图 7.2　一个标签的外观

7.3.2　TextBox 控件

TextBox 控件又称文本框控件。TextBox 控件用于让用户输入文本字符串数据，它的常用属性、方法和事件如表 7.4 所示。

表 7.4　TextBox 控件的常用属性、方法和事件

类型	名　　称	说　　明
属性	AutoPostBack	获取或设置一个值，该值表示控件失去焦点时是否发生自动回发到服务器的操作
	Columns	设置文本框的水平尺寸，单位为字符
	MaxLength	设置文本框的最大字符数(不能用于多行文本框)
	ReadOnly	设置文本框是否为只读的
	Rows	设置文本框的垂直尺寸
	Text	设置文本框中显示的文本
	TextMode	设置文本框的文本模式，可取 MultiLine(多行)、Password(作为密码输入)或 SingleLine(单行)等值
	Wrap	设置多行文本框中的文本是否回绕
方法	OnTextChanged	引发 TextChanged 事件
事件	TextChanged	当文本框内容发生改变时引发此事件

默认情况下，TextMode 属性设置为 SingleLine，它创建只包含一行的文本框。还可将此属性设置为 MultiLine 或 Password，分别用于多行文本输入和密码输入。在作为密码输入文本框时，用户的输入用“*”替代，达到保密的效果。文本框的显示宽度由其 Columns 属性确定。如果文本框是多行文本框，则其显示高度由 Rows 属性确定。

如图 7.3 所示，从左到右分别是 TextMode 属性为 SingleLine、Password 和 MultiLine (Rows=3)的 3 个文本框。

China ●●●●● China

图 7.3 3 种不同类型的文本框

使用 Text 属性获取用户在 TextBox 控件中输入的内容。通过设置 MaxLength 属性，可以限制可输入到此控件中的字符数。将 Wrap 属性设置为 true 来指定当到达文本框的结尾时，其中的内容应自动在下一行继续。

AutoPostBack 属性决定控件中文本内容修改后，是否自动回发到服务器。默认为 False，即修改文本后并不立即回发到服务器，而是等网页表单被提交后一并处理。若为 True，则每次更改文本框的内容并且焦点离开控件时，都会自动回发，并使服务器处理该控件相应的 TextChanged 事件。

7.3.3 Button 控件

Button 控件又称命令按钮控件，它的常用属性、方法和事件如表 7.5 所示。

表 7.5 Button 控件的常用属性、方法和事件

类　型	名　称	说　明
属性	Text	设置命令按钮上显示的文本
	CommandName	单击命令按钮时，该值来指定一个命令名称
	CommandArgument	单击命令按钮时，将该值来传递给 Command 事件
	CausesValidation	设置为 False，则所提交作为参数的表单不被检验，默认为 True
方法	OnClick	引发 Click 事件
	OnCommand	引发 Command 事件
事件	Click	单击命令按钮且包含它的表单被提交到服务器时，引发此事件
	Command	单击命令按钮时，引发此事件

默认的 Button 按钮为 Submit(提交)按钮，这种情况下不要指定 CommandName 属性和 CommandArgument 属性值，其功能是在单击时激活 Click 事件将包含它的表单提交给相应的服务器进行处理。

当设置 CommandName 属性和 CommandArgument 属性后，Button 按钮成为 Command (命令)按钮，在单击时激活 Command 事件。当多个 Command 按钮共用一个 OnCommand 方法时，可以根据 CommandArgument 值确定单击了哪个 Button 控件。对于 Command 按钮，在单击时激活 Command 事件，同样会将包含它的表单提交给相应的服务器进行处理。

例如，在一个网页中设计如下两个 Command 按钮 Button1 和 Button2：

```
<asp:Button ID = "Button1" runat = "server" CommandArgument = "第 1 个 Command 按钮"
    CommandName = "Command" OnCommand = "Button_Command" Text = "命令按钮 1" />
<asp:Button ID = "Button2" runat = "server" CommandArgument = "第 2 个 Command 按钮"
    CommandName = "Command" OnCommand = "Button_Command" Text = "命令按钮 2" />
```

设计 Button_Command 事件处理方法如下：

```
protected void Button_Command(object sender, CommandEventArgs e)
{
    Label1.Text = e.CommandArgument.ToString();
}
```

在执行该网页时，用户单击“命令按钮 1”，在 Label1 标签中显示“第 1 个 Command 按钮”，用户单击“命令按钮 2”，在 Label1 标签中显示“第 2 个 Command 按钮”。

另外，许多 Web 服务器控件都有 Attributes 属性，它是一个集合(AttributeCollection 类对象)，用于获取在 ASP .NET 网页内的服务器控件标记上表示的所有属性名称和值对的集合。Attributes 属性使开发人员可以采用编程方式控制与 Web 服务器控件关联的属性。例如：

```
Button1.Attributes.Add("onclick", "alert('OK');return false");
```

该语句的功能是：在 ASP .NET 将 Button1 控件转换为客户端的 HTML 标记之后，在该元素上添加一个 onclick 属性，用户单击时执行 JavaScript 脚本代码“alert('OK');return false”。

又如：

```
TextBox1.Attributes.Add("value", TextBox2.Text);
```

该语句的功能是：给客户端的 TextBox1 文本框添加 value 属性，用于显示 TextBox2 文本框的值。

上述语句通常放在网页的 Page_Load 事件处理方法中。

7.3.4 Image 控件

Image 控件又称图像控件，用于在网页上显示图像，它的常用属性如表 7.6 所示。

表 7.6　Image 控件的常用属性

属　性	说　明
AlternateText	获取或设置当图像不可用时，Image 控件中显示的替换文本
ImageAlign	获取或设置 Image 控件相对于网页上其他元素的对齐方式
ImageUrl	获取或设置在 Image 控件中显示的图像的位置

ImageUrl 属性用来获取 Image 控件中要显示的图像的地址，在通过“属性”窗口设置该属性时，单击 ImageUrl 属性后的[...]按钮，弹出一个“选择图像”对话框，可以从中选择要显示的图像。

例如，如下 Image1 控件显示 stud.jpg 图像：

```
<asp:Image ID = "Image1" runat = "server" ImageUrl = "~/Image/stud.jpg" />
```

此外，还有 ImageButton 控件，其使用与 Image 控件类似。

7.3.5 HyperLink 控件

HyperLink 控件又称超链接控件，相当于实现了 HTML 代码中的“”效果。HyperLink 控件可以通过传递指定的参数来访问不同的网页。当引发一个事件后，超链接的属性可以被改变。HyperLink 控件的常用属性如表 7.7 所示。

例如，单击如下 HyperLink1 控件时转向 webform1.aspx 网页：

```
<asp:HyperLink ID = "HyperLink1" runat = "server" NavigateUrl = "~/webform1.aspx">
   转向 webform1
</asp:HyperLink>
```

表 7.7　HyperLink 控件的常用属性

属　性	说　明
ImageUrl	获取或设置为 HyperLink 控件显示的图像的路径
NavigateUrl	获取或设置单击 HyperLink 控件时链接到的 URL
Target	指定 NavigateUrl 的目标框架
Text	获取或设置 HyperLink 控件的文本标题

此外，还有 LinkButton 控件，以命令按钮的方式显示超链接，其使用与 HyperLink 控件类似。

7.3.6　CheckBox 控件

CheckBox 控件又称复选框控件，对应命名空间 System. Web. UI. WebControls 中的 CheckBox 类。该控件为用户提供了一种输入布尔型数据的方法，允许用户进行多项选择，对应 HTML 的＜input type＝"checkbox"＞。CheckBox 控件的常用属性和事件如表 7.8 所示。

表 7.8　CheckBox 控件的常用属性和事件

类型	名称	说　明
属性	AutoPostBack	获取或设置一个值，该值指示在单击时 CheckBox 状态是否自动回发到服务器
	Checked	获取或设置一个值，该值指示是否已选中 CheckBox 控件
	Text	获取或设置与 CheckBox 关联的文本标签
事件	CheckedChanged	当 Checked 属性的值在向服务器进行发送期间更改时发生

例如，如下代码在网页中放置 3 个 CheckBox 控件：

```
<asp:CheckBox ID = "CheckBox1" runat = "server" Text = "选项 1" />
<asp:CheckBox ID = "CheckBox2" runat = "server" Text = "选项 2" />
<asp:CheckBox ID = "CheckBox3" runat = "server" Text = "选项 3" />
```

7.3.7　RadioButton 控件

RadioButton 控件又称单选按钮控件。RadioButton 类派生自 CheckBox 类，允许用户互斥地从多个 RadioButton 控件中选择一个。RadioButton 控件的常用属性和事件如表 7.9 所示。

表 7.9　RadioButton 控件的常用属性和事件

类型	名称	说　明
属性	AutoPostBack	获取或设置一个值，该值指示在单击时 RadioButton 状态是否自动回发到服务器
	Checked	获取或设置一个值，该值指示是否已选中 RadioButton 控件
	GroupName	获取或设置单选按钮所属的组名
	Text	获取或设置与 RadioButton 关联的文本标签
事件	CheckedChanged	当 Checked 属性的值在向服务器进行发送期间更改时发生

RadioButton 控件中 GroupName 是一个非常重要的属性，如果网页中有多个 RadioButton 控件，那些 GroupName 属性相同的 RadioButton 控件在逻辑上属于一个组，所以对于属同一组的 RadioButton 控件需将它们 GroupName 属性设置为同一值。

RadioButton 控件主要的事件是 CheckedChanged，当用户单击该控件时引发执行对应的事件处理方法。

例如，如下代码在网页中放置 3 个互斥的 RadioButton 控件：

```
< asp:RadioButton ID = "RadioButton1" runat = "server" GroupName = "xm" Text = "选项 1" />
< asp:RadioButton ID = "RadioButton2" runat = "server" GroupName = "xm" Text = "选项 2" />
< asp:RadioButton ID = "RadioButton3" runat = "server" GroupName = "xm" Text = "选项 3" />
```

【练一练】 以“D:\电子商务\CH7”目录创建文件系统空网站 CH7，添加一个 webform1 网页，其功能是说明常用的表单控件的使用方法。

其设计步骤如下：

① 打开 CH7 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH7”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform1. aspx，其他保持默认项，单击“添加”按钮。

② 本网页的设计界面如图 7.4 所示，有一个 7 行 2 列的表格，每行放置若干相应的 Web 标准控件。



图 7.4　webform1 网页的设计界面

对应的源视图代码如下：

```
<%@ Page Language = "C#" AutoEventWireup = "true" CodeFile = "webform1.aspx.cs"
    Inherits = "webform1" %>
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head runat = "server">
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8"/>
    <title></title>
    <style type = "text/css">
        .auto-capstyle {
            font-family: 楷体; font-size: medium;
            color: #0000FF;     font-weight: bold;
            text-align:right;
        }
        .auto-rcstyle {
            font-family: 仿宋;  font-weight: bold;
            font-size: medium;  color: #008000;
        }
    </style>
  </head>
  <body style = "width: 400px">
```

```
<form id = "form1" runat = "server">
    <table>
      <tr>
        <td colspan = "2" style = "text - align:center">
          <asp:Image ID = "Image1" runat = "server" ImageUrl = "～/Image/stud.jpg" />
        </td>
      </tr>
      <tr>
        <td style = "width:100px" class = "auto - capstyle">学号</td>
        <td><asp:TextBox ID = "TextBox1" runat = "server"
               style = "width:100px"></asp:TextBox>
        </td>
      </tr>
      <tr>
        <td class = "auto - capstyle" >姓名</td>
        <td><asp:TextBox ID = "TextBox2" runat = "server"
               style = "width:100px"></asp:TextBox>
        </td>
      </tr>
      <tr>
        <td class = "auto - capstyle" >性别</td>
        <td><asp:RadioButton ID = "RadioButton1" runat = "server" Text = "男"
               CssClass = "auto - rcstyle" GroupName = "sex" />
             
            <asp:RadioButton ID = "RadioButton2" runat = "server" Text = "女"
               CssClass = "auto - rcstyle" GroupName = "sex" />
        </td>
      </tr>
      <tr>
        <td class = "auto - capstyle" >爱好</td>
        <td ><asp:CheckBox ID = "CheckBox1" runat = "server"
                CssClass = "auto - rcstyle" Text = "打球" />
             <asp:CheckBox ID = "CheckBox2" runat = "server"
                CssClass = "auto - rcstyle" Text = "跑步" />
             <asp:CheckBox ID = "CheckBox3" runat = "server"
                CssClass = "auto - rcstyle" Text = "看书" />
             <asp:CheckBox ID = "CheckBox4" runat = "server"
                CssClass = "auto - rcstyle" Text = "上网" />
        </td>
      </tr>
      <tr>
      <td colspan = "2" style = "text - align:center">
           <asp:Button ID = "Button1" runat = "server" Text = "确定"
             style = "color: ＃FF0000; font - size: medium; font - weight: 700;
               font - family: 黑体" OnClick = "Button1_Click" />
      </td>
    </tr>
    <tr>
      <td colspan = "2" style = "color: ＃FF00FF; font - size: medium; font - weight: 700;
          font - family: 仿宋">
          <asp:Label ID = "Label1" runat = "server"></asp:Label>
      </td>
    </tr>
  </table>
</form>
```

```
  </body>
</html>
```

③ 在该网页上设计如下事件过程：

```
protected void Button1_Click(object sender, EventArgs e)
{   string mystr = "你的个人信息如下:<br>";
    if (TextBox1.Text == "")
    {   Label1.Text = "提示: 必须输入学号";
        return;
    }
    else  mystr += " 学号为" + TextBox1.Text + "<br>";
    if (TextBox2.Text == "")
    {   Label1.Text = "提示: 必须输入姓名";
        return;
    }
    else  mystr += " 姓名为" + TextBox2.Text + "<br>";
    if (RadioButton1.Checked)
        mystr += " 性别为男<br>";
    if (RadioButton2.Checked)
        mystr += " 性别为女<br>";
    if (CheckBox1.Checked)
        mystr += " 爱好有" + CheckBox1.Text;
    if (CheckBox2.Checked)
        mystr += " 爱好有" + CheckBox2.Text;
    if (CheckBox3.Checked)
        mystr += " 爱好有" + CheckBox3.Text;
    if (CheckBox4.Checked)
        mystr += " 爱好有" + CheckBox4.Text;
    mystr += "<br>";
    Label1.Text = mystr;
}
```

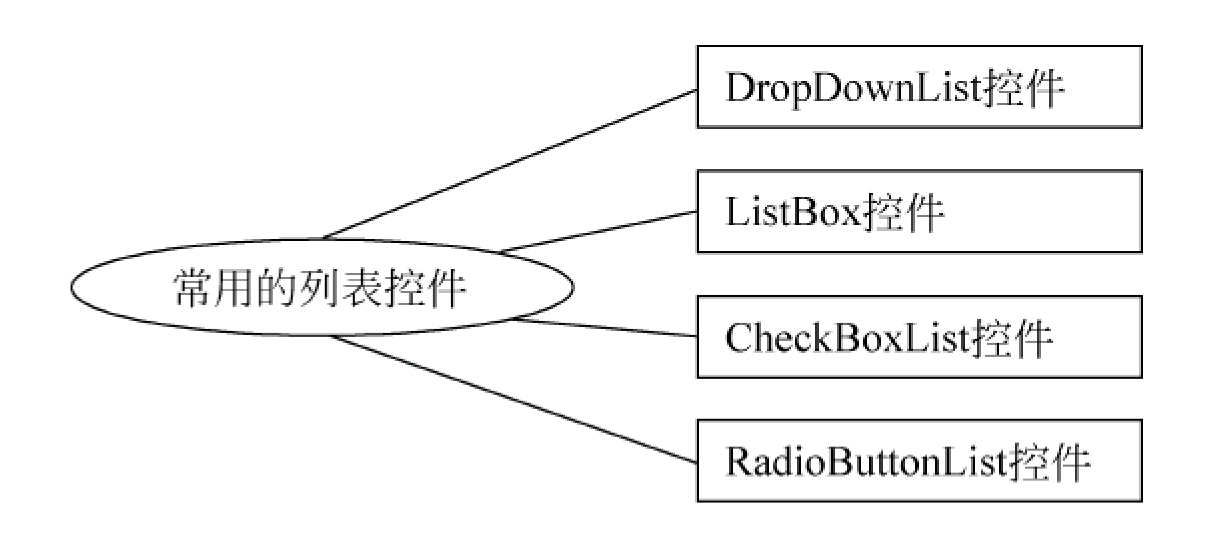
④ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，进行相应操作，单击“确定”按钮，其结果如图 7.5 所示。



图 7.5　webform1 网页的执行界面

7.4 常用的列表控件

知识梳理



7.4.1 DropDownList 控件

DropDownList 控件又称下拉列表控件，它是从 ListControl 类派生的。ListControl 类用作定义所有列表类型控件通用的属性、方法和事件的抽象基类，所以所有的列表控件具有许多相同的特性。

使用 DropDownList 控件可以创建只允许从中选择一项的下拉列表控件，它的常用属性和事件如表 7.10 所示。

表 7.10 DropDownList 控件的常用属性和事件

类型	名　　称	说　　明
属性	AutoPostBack	指示当用户更改列表中的选定内容时是否自动产生向服务器的回发
	DataMember	获取或设置要绑定到控件的 DataSource 中的特定表
	DataSource	获取或设置填充列表控件项的数据源
	DataTextField	获取或设置为列表项提供文本内容的数据源字段
	DataValueField	获取或设置为各列表项提供值的数据源字段
	Items	获取列表控件项的集合
	SelectedItem	获取列表控件中索引最小的选定项
事件	SelectedIndexChanged	当列表控件的选定项在信息发往服务器之间变化时发生
	TextChanged	当 Text 和 SelectedValue 属性更改时发生

使用 SelectedIndex 属性以编程方式指定或确定 DropDownList 控件中的选定项的索引。DropDownList 控件中总是选择一项，无法同时取消选择列表中的所有项。DropDownList 控件中的项的索引从零开始。

注意： 若要 DropDownList 控件执行 SelectedIndexChanged 等事件处理方法，需要将该控件的 AutoPostBack 属性设置为 True(默认值为 False)。也就是说，只要用户从列表控件中进行选择，就会立即引发 SelectedIndexChanged 事件，如果 AutoPostBack 属性为 Ttrue，则每次选择时都将表单发送到服务器，但在每个往返行程中选定的项保持不变。

DropDownList 控件的 Items 是一个集合属性，其中每个元素(项)是一个 ListItem 对象。Items 是 ListItemCollection 类的对象，而 ListItemCollection 类的常用属性和方法如表 7.11 所示。Items 集合属性的示意图如图 7.6 所示。

表 7.11　ListItemCollection 类的常用属性和事件

类　　型	名　　称	说　　明
属性	Count	获取集合中的 ListItem 对象数
	Item	获取集合中指定索引处的 ListItem
方法	Add	将 ListItem 追加到集合的结尾
	AddRange	将 ListItem 对象数组中的项添加到集合
	Clear	从集合中移除所有 ListItem 对象
	Contains	确定集合是否包含指定的项
	FindByText	搜索集合中具有 Text 属性且包含指定文本的 ListItem
	FindByValue	搜索集合中具有 Value 属性且包含指定值的 ListItem
	IndexOf	确定索引值，该值表示指定 ListItem 在集合中的位置
	Insert	将 ListItem 插入集合中的指定索引位置
	Remove	从集合中移除 ListItem
	RemoveAt	从集合中移除指定索引位置的 ListItem



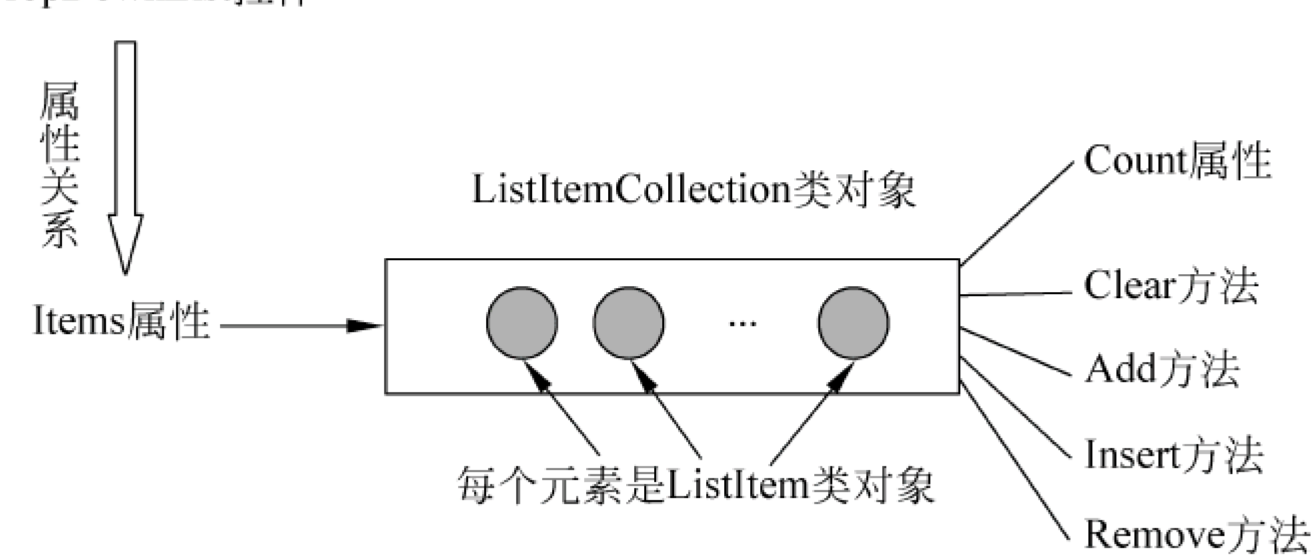


图 7.6　Items 集合属性的示意图

注意：在 ASP .NET 中，许多控件或对象都有集合属性，这些集合属性的属性和方法与 ListItemCollection 类的相似，在使用方式上与 ListItemCollection 类的相同。

在设计时设置 Items 属性的方法是，单击 DropDownList1 控件右上角的按钮，出现“DropDownList 任务”菜单，选择“编辑项”命令，打开“ListItem 集合编辑器”对话框，通过“添加”按钮增加 Items 属性中的各个项，并输入各项的 Text 和 Value 属性值。

在运行程序时也可以动态地向 DropDownList 控件中添加项，例如，以下语句用于向 DropDownList1 控件中添加 4 个项：

```
DropDownList1.Items.Add("打球");
DropDownList1.Items.Add("跑步");
DropDownList1.Items.Add("看书");
DropDownList1.Items.Add("上网");
```

7.4.2　ListBox 控件

ListBox 控件又称下拉列表框控件。ListBox 控件允许用户从预定义列表中选择一项或多项。它与 DropDownList 控件不同之处在于它可一次显示多项，也可允许用户选择多项。ListBox 控件的常用属性和事件如表 7.12 所示。

表 7.12 ListBox 控件的常用属性和事件

类型	名　称	说　明
属性	AutoPostBack	指示当用户更改列表中的选定内容时是否自动产生向服务器的回发
	DataMember	获取或设置数据绑定控件绑定到的数据列表的名称
	DataSource	获取或设置填充 ListBox 控件的数据源
	DataTextField	获取或设置为 ListBox 控件提供文本内容的数据源字段
	DataValueField	获取或设置为各列表项提供值的数据源字段
	Items	获取 ListBox 控件项的集合
	Rows	获取或设置 ListBox 控件中显示的行数
	SelectedItem	获取 ListBox 控件中索引最小的选定项
	SelectedMode	获取或设置 ListBox 控件的选择模式，可选 Single（只能选一项）或 Multiple（可以选多项）
事件	SelectedIndexChanged	当列表控件的选定项在信息发往服务器之间变化时发生
	TextChanged	当 Text 和 SelectedValue 属性更改时发生

ListBox 控件的 Items 是一个集合属性，其中每个元素（项）是一个 ListItem 对象。Items 属性用来设置子选项；每个子选项都具有索引值，索引值开始值为 0。实际上，ListBox 控件 Items 属性也是一个 ListItemCollection 类对象，ListItemCollection 类的常用属性和方法如表 7.16 所示。

向 ListBox 控件中添加项的过程与 DropDownList 控件的过程相似。与 DropDownList 控件相同，若要执行 SelectedIndexChanged 等事件处理方法，需要将该控件的 AutoPostBack 属性设置为 True（默认值为 False）。另外，若允许多行选择，需要将 SelectionMode 设置为 Multiple。

7.4.3 CheckBoxList 控件

CheckBoxList 控件又称为复选框列表控件。CheckBoxList 控件与 CheckBox 控件类似，不同之处是前者只有一个复选框，后者可以包含多个复选框。它的常用属性和事件如表 7.13 所示。

表 7.13 CheckBoxList 控件的常用属性和事件

类型	名　称	说　明
属性	AutoPostBack	获取或设置一个值，该值指示当用户更改列表中的选定内容时是否自动产生向服务器的回发
	Items	表示控件对象中所有项的集合
	SelectedIndex	获取或设置列表中选定项的最低序号索引
	SelectedItem	获取列表控件中索引最小的选定项
	SelectedValue	获取列表控件中选定项的值，或选择列表控件中包含指定值的项
	Text	获取或设置 CheckBoxList 控件的 SelectedValue 属性
	RepeatColumns	获取或设置要在 CheckBoxList 控件中显示的列数
	RepeatDirection	获取或设置一个值，该值指示控件是垂直显示还是水平显示，取值与 RadioButtonList 控件的该属性相同
	RepeatLayout	获取或设置一个值，该值指定是否将使用 table 元素（默认值）、ul 元素、ol 元素或 span 元素来呈现列表
	TextAlign	与 RadioButtonList 控件的该属性相同
事件	CheckedIndexChanged	当列表控件的选定项在信息发往服务器之间变化时发生
	TextChanged	当 Text 和 SelectedValue 属性更改时发生

CheckBoxList 控件中 Items 属性用来设置子选项；每个子选项都是一个 ListItem 对象，都具有索引值，索引值开始值为 0。Items 是 ListItemCollection 类的对象。CheckBoxList 控件中使用一组 Selected 属性来判断子选项是否被选中。若要确定 CheckBoxList 控件中的选定项，可以循环访问 Items 集合并测试该集合中每一项的 Selected 属性。

和 ListBox 一样，可以通过“ListItem 集合编辑器”对话框添加 CheckBoxList 控件的各个选项。同样，CheckBoxList 控件的主要事件是 SelectedIndexChanged，当用户单击其中的一个选项时引发执行对应的事件处理方法，但需要设置其 AutoPostBack 属性为 True。

默认情况下，CheckBoxList 控件只显示一列按钮。开发人员可以将 CheckBoxList 控件的 RepeatColumns 属性设置为所需的列数。在这些列中，还可以使用 RepeatDirection 枚举将 RepeatDirection 属性设置为 Vertical(默认值)或 Horizontal。

7.4.4　RadioButtonList 控件

RadioButtonList 控件又称单选按钮列表控件。RadioButtonList 控件用于构建单选按钮列表，允许用户互斥地其列表中选择一个。RadioButtonList 控件的常用属性和事件如表 7.14 所示。

表 7.14　RadioButtonList 控件的常用属性和事件

类型	名　称	说　明
属性	AutoPostBack	获取或设置一个值，该值指示当用户更改列表中的选定内容时是否自动产生向服务器的回发
	Items	表示控件对象中所有项的集合
	SelectedIndex	获取或设置列表中选定项的最低序号索引
	SelectedItem	获取列表控件中索引最小的选定项
	SelectedValue	获取列表控件中选定项的值，或选择列表控件中包含指定值的项
	Text	获取或设置 RadioButtonList 控件的 SelectedValue 属性
	RepeatColumns	获取或设置要在 RadioButtonList 控件中显示的列数
	RepeatDirection	获取或设置组中单选按钮的显示方向。为 Horizontal 表示列表项以行的形式水平显示，从左到右、自上而下地加载，直到呈现出所有的项。为 Vertical(默认值)表示列表项以列的形式垂直显示，自上而下、从左到右地加载，直到呈现出所有的项
	RepeatLayout	获取或设置一个值，该值指定是否将使用 table 元素(默认值)、ul 元素、ol 元素或 span 元素来呈现列表
	TextAlign	获取或设置组内单选按钮的文本对齐方式，取值为 Left 表示与单选按钮控件关联的文本显示在该控件的左侧；取值为 Right(默认值)表示与单选按钮控件关联的文本显示在该控件的右侧
事件	CheckedIndexChanged	当列表控件的选定项在信息发往服务器之间变化时发生
	TextChanged	当 Text 和 SelectedValue 属性更改时发生

RadioButtonList 控件中 Items 属性来设置子选项；每个子选项都具有索引值，索引值开始值为 0；Items 是 ListItemCollection 类的对象，而 ListItemCollection 类的常用属性和方法见表 7.11。RadioButtonList 控件中使用一组 Selected 属性来判断子选项是否被选中。

和 ListBox 一样，可以通过“ListItem 集合编辑器”对话框添加 RadioButtonList 控件的各个选项。同样，RadioButtonList 控件的主要事件是 SelectedIndexChanged，当用户单击其中

的一个选项时引发执行对应的事件处理方法，但需要设置其 AutoPostBack 属性为 True。

默认情况下，RadioButtonList 控件只显示一列按钮。开发人员可以将 RadioButtonList 控件的 RepeatColumns 属性设置为所需的列数。在这些列中，还可以使用 RepeatDirection 枚举将 RepeatDirection 属性设置为 Vertical(默认值)或 Horizontal。

【练一练】 在本章网站 CH7 中添加一个 webform2 网页，其功能是说明常用的列表控件的使用方法。

其设计步骤如下：

① 打开 CH7 网站，选择"网站|添加新项"菜单命令，出现"添加新项-CH7"对话框，在中间列表中选择"Web 窗体"，将文件名称改为 webform2. aspx，其他保持默认项，单击"添加"按钮。

② 本网页的设计界面有一个 8 行 2 列的表格，每行放置若干相应的 Web 标准控件，如图 7.7 所示。

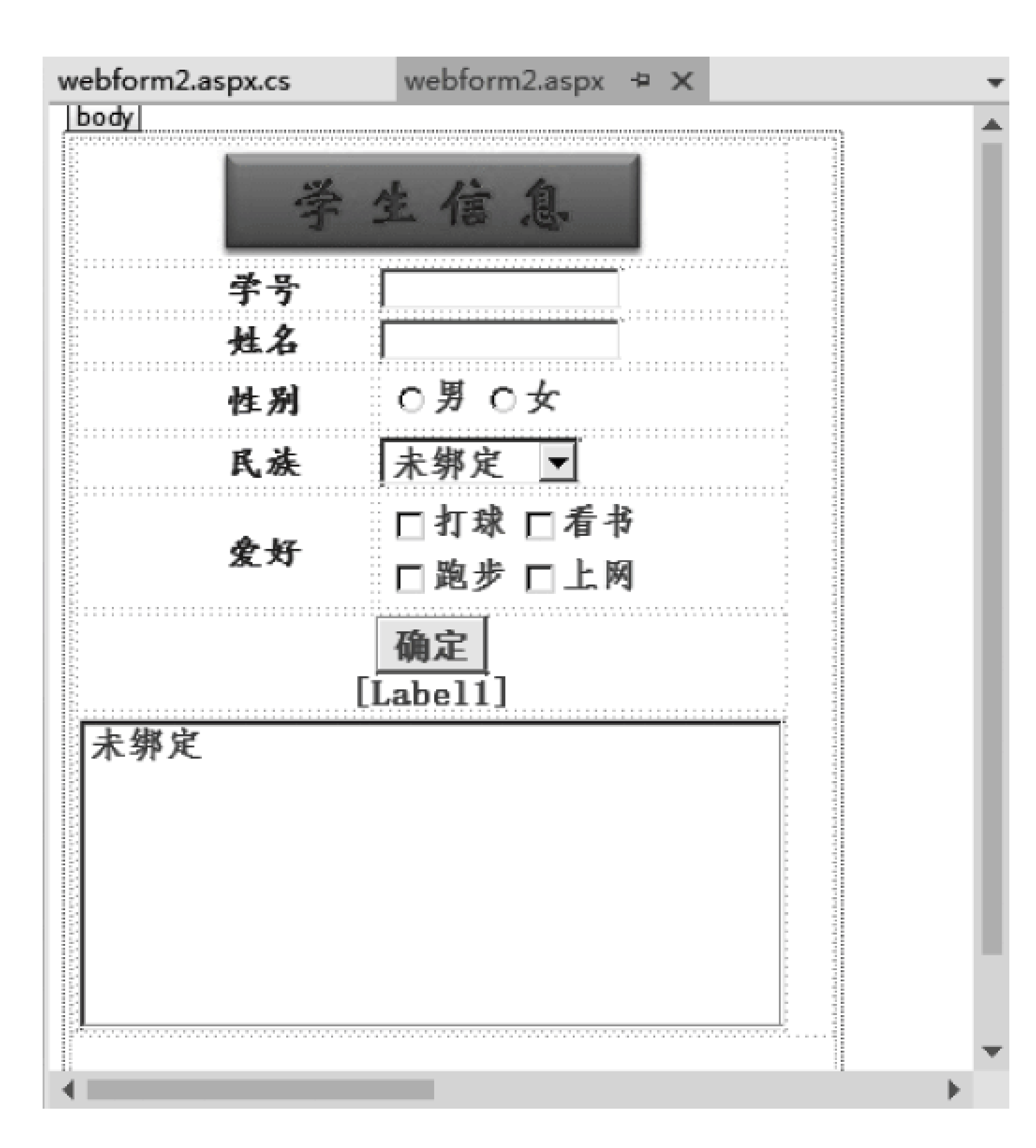


图 7.7 webform2 网页的设计界面

对应的源视图代码如下：

```
<%@ Page Language = "C#" AutoEventWireup = "true" CodeFile = "webform2.aspx.cs"
    Inherits = "webform2" %>
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head id = "Head1" runat = "server">
    <meta http - equiv = "Content - Type" content = "text/html; charset = utf - 8"/>
    <title></title>
    <style type = "text/css">
        .auto - capstyle {
            font - family: 楷体; font - size: medium;
            color: #0000FF; font - weight: bold;
            width:100px; text - align:right;
        }
        .auto - rcstyle {
            font - family: 仿宋; font - weight: bold;
            font - size: medium; color: #008000;
        }
```

```
    </style>
</head>
<body style="width: 341px">
    <form id="form1" runat="server">
        <table>
            <tr>
                <td colspan="2" style="text-align:center">
                    <asp:Image ID="Image1" runat="server" ImageUrl="~/Image/stud.jpg" />
                </td>
            </tr>
            <tr>
                <td class="auto-capstyle">学号</td>
                <td>
                    <asp:TextBox ID="TextBox1" runat="server" style="width:100px" />
                </td>
            </tr>
            <tr>
                <td class="auto-capstyle" >姓名</td>
                <td>
                    <asp:TextBox ID="TextBox2" runat="server" style="width:100px" />
                </td>
            </tr>
            <tr>
                <td class="auto-capstyle" >性别</td>
                <td>
                    <asp:RadioButtonList ID="RadioButtonList1" runat="server"
                        RepeatDirection="Horizontal" CssClass="auto-rcstyle">
                        <asp:ListItem Value="男" ></asp:ListItem>
                        <asp:ListItem>女</asp:ListItem>
                    </asp:RadioButtonList>
                </td>
            </tr>
            <tr>
                <td class="auto-capstyle">民族</td>
                <td>
                    <asp:DropDownList ID="DropDownList1" runat="server" Height="16px"
                        Width="89px" CssClass="auto-rcstyle">
                    </asp:DropDownList>
                </td>
            </tr>
            <tr>
                <td class="auto-capstyle">爱好</td>
                <td>
                    <asp:CheckBoxList ID="CheckBoxList1" runat="server" RepeatColumns="2"
                        CssClass="auto-rcstyle">
                        <asp:ListItem Value="打球"></asp:ListItem>
                        <asp:ListItem Value="跑步"></asp:ListItem>
                        <asp:ListItem>看书</asp:ListItem>
                        <asp:ListItem>上网</asp:ListItem>
                    </asp:CheckBoxList>
                </td>
            </tr>
            <tr>
                <td colspan="2" style="text-align:center">
                    <asp:Button ID="Button1" runat="server" Text="确定"
```

```
                    style = "color: #FF0000; font - size: medium; font - weight: 700;
                      font - family: 黑体" OnClick = "Button1_Click" />
                <br />
                <asp:Label ID = "Label1" runat = "server"
                   style = "color: #FF00FF; font - size: medium; font - weight: 700;
                    font - family: 仿宋" />
              </td>
            </tr>
            <tr>
              <td colspan = "2" style = "color: #FF00FF; font - size: medium;
                  font - weight:700; font - family: 仿宋">
                 <asp:ListBox ID = "ListBox1" runat = "server" Width = "316px"
                   style = "color: #FF00FF; font - size: medium;
                     font - weight: 700; font - family: 仿宋" Height = "140px" />
              </td>
            </tr>
         </table>
      </form>
   </body>
</html>
```

③ 在该网页上设计如下事件过程：

```
protected void Page_Load(object sender, EventArgs e)
{   DropDownList1.Items.Add("汉族");
    DropDownList1.Items.Add("回族");
    DropDownList1.Items.Add("满族");
    DropDownList1.Items.Add("其他");
}
protected void Button1_Click(object sender, EventArgs e)
{   ListBox1.Items.Clear();
    ListBox1.Items.Add("你的个人信息如下:");
    if (TextBox1.Text == "")
    {   Label1.Text = "提示：必须输入学号";
        return;
    }
    else ListBox1.Items.Add("学号为" + TextBox1.Text);
    if (TextBox2.Text == "")
    {   Label1.Text = "提示：必须输入姓名";
        return;
    }
    else ListBox1.Items.Add("姓名为" + TextBox2.Text);
    foreach (ListItem it in RadioButtonList1.Items)
        if (it.Selected == true)
            ListBox1.Items.Add("性别为" + it.Text);
    if (DropDownList1.SelectedValue != "")
        ListBox1.Items.Add("民族为" + DropDownList1.SelectedValue);
    string mystr = "";
    foreach (ListItem it in CheckBoxList1.Items)
        if (it.Selected == true)
           mystr += it.Text + " ";
    ListBox1.Items.Add("爱好为:" + mystr);
}
```

④ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，进行相应操作，单击“确定”命令按钮，其结果如图 7.8 所示。

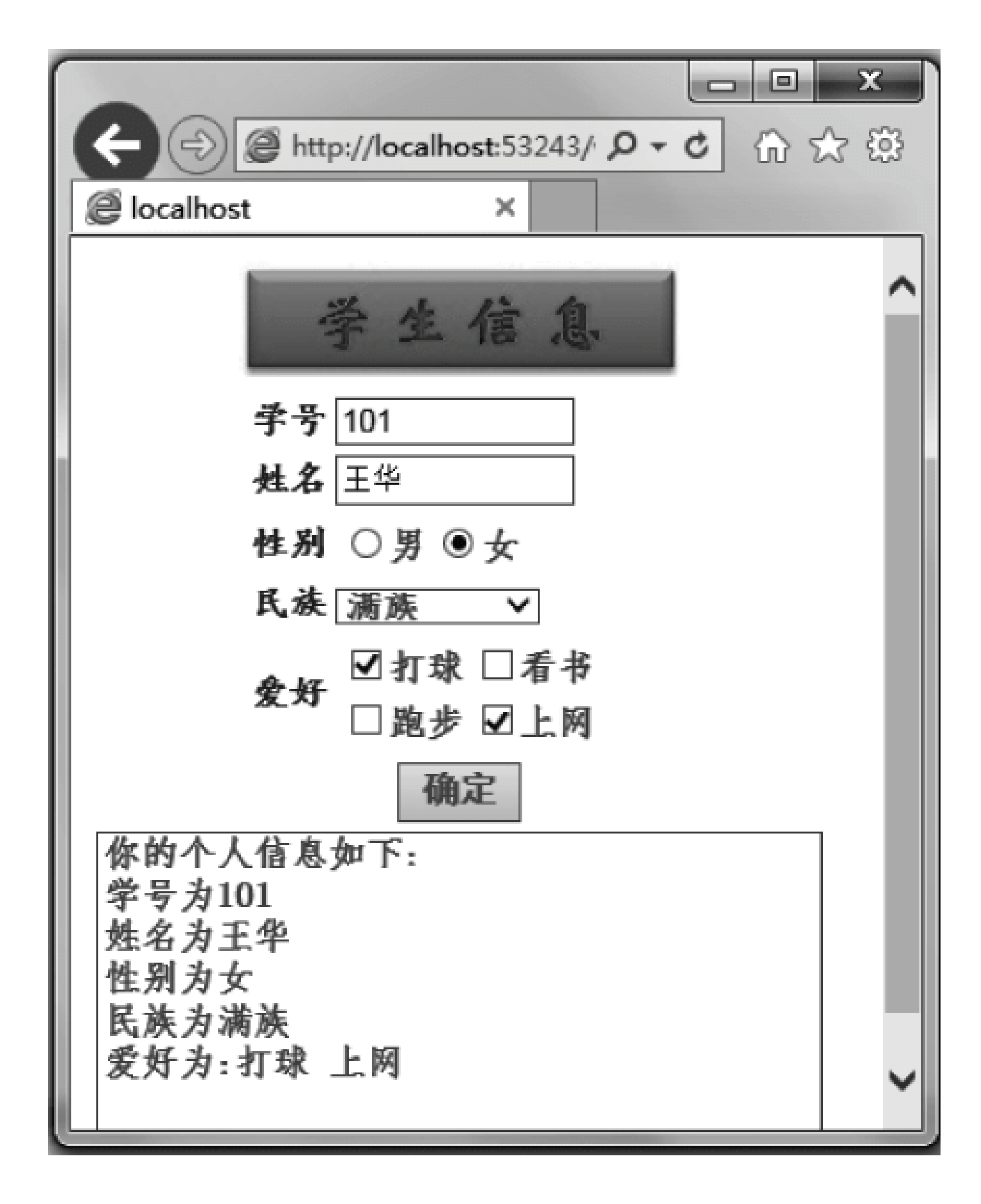
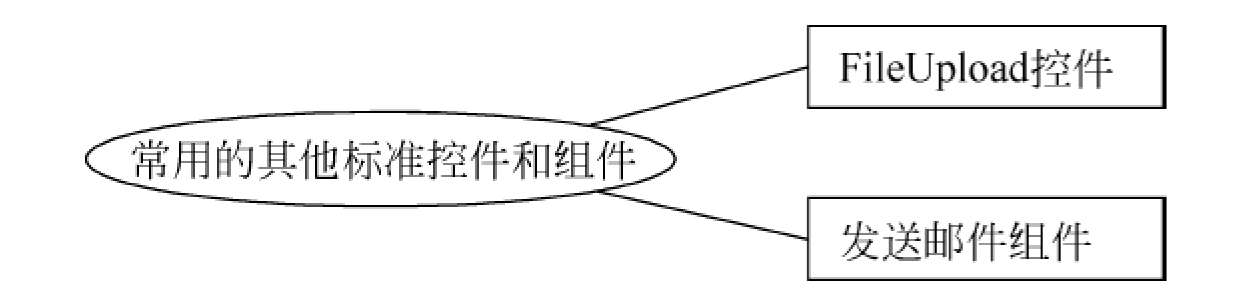


图 7.8　webform2 网页的执行界面

7.5　常用的其他标准控件和组件

知识梳理



7.5.1　FileUpload 控件

FileUpload 控件又称上传文件控件，它显示一个文本框控件和一个浏览按钮，使用户可以选择要上传到服务器的文件。其常用属性和方法如表 7.15 所示。

表 7.15　FileUpload 控件的常用属性和方法

类型	名　称	说　明
属性	FileBytes	从使用 FileUpload 控件指定的文件返回一个字节数组
	FileContent	获取 Stream 对象，它指向要使用 FileUpload 控件上载的文件
	FileName	获取客户端上使用 FileUpload 控件上载的文件的名称
	HasFile	获取一个值，该值指示 FileUpload 控件是否包含文件
	PostedFile	获取使用 FileUpload 控件上载的文件的 HttpPostedFile 对象
方法	SaveAs	使用 FileUpload 控件将上载的文件的内容保存到 Web 标准服务器上的指定路径

FileUpload 控件的 PostedFile 属性是 HttpPostedFile 类对象，而 HttpPostedFile 类提供对客户端已上载的单独文件的访问。其常用属性和方法如表 7.16 所示。

表 7.16　HttpPostedFile 类的常用属性和方法

类型	名　　称	说　　明
属性	ContentLength	获取上载文件的大小(以字节为单位)
	ContentType	获取客户端发送的文件的 MIME 内容类型
	FileName	获取客户端上的文件的完全限定名称
	InputStream	获取一个 Stream 对象，该对象指向一个上载文件，以准备读取该文件的内容
方法	SaveAs	保存上载文件的内容

在用户选择要上载的文件后，FileUpload 控件不会自动将该文件保存到服务器。必须显式提供一个控件或机制，使用户能提交指定的文件。例如，可以提供一个命令按钮，用户单击它即可上载文件。为保存指定文件应调用 SaveAs 方法，该方法将文件内容保存到服务器上的指定路径。

【练一练】 在本章网站 CH7 中添加一个 webform3 网页，其功能是说明 FileUpload 控件的使用方法。

其设计步骤如下：

① 打开 CH7 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH7”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform3. aspx，其他保持默认项，单击“添加”按钮。

② 在 CH7 网站中建立一个存放上传文件的子目录，如 File。

③ 本网页的设计界面如图 7.9 所示，有一个 FileUpload 控件 FileUpload1(由一个文本框和“浏览”按钮组成)、一个命令按钮 Button1 和一个标签 Label1。在该网页上设计如下事件过程：

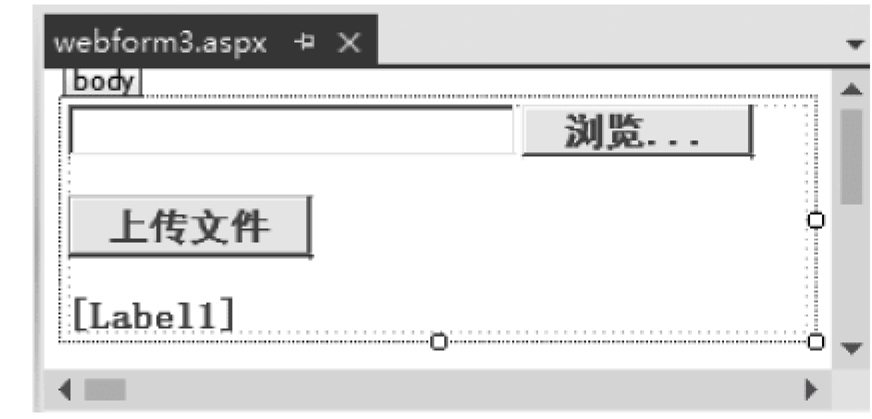


图 7.9　webform3 网页的设计界面

```
protected void Button1_Click(object sender, EventArgs e)
{   string filestr;
    if (FileUpload1.HasFile)
    {   filestr = Server.MapPath("") + "\\File\\" + FileUpload1.PostedFile.FileName;
        try
        {   FileUpload1.PostedFile.SaveAs(filestr);
            Label1.Text = "提示: 文件成功上传到" + filestr;
        }
        catch (Exception ex)
        {
            Label1.Text = "提示: 文件上传失败," + ex.Message;
        }
    }
    else
        Label1.Text = "提示: 没有指定任何要上传的文件";
}
```

④ 单击工具栏中的 ▶ Internet Explorer 按钮运行本网页，单击“浏览”按钮，在出现的“选择要加载的文件”对话框中选中一个要上传的文件 1111.jpg，如图 7.10 所示，单击“上传文件”命令按钮，将指定的文件复制到“D:\电子商务\CH7\File”目录中，其结果如图 7.11 所示。




图 7.10　webform3 网页的执行界面一



图 7.11　webform3 网页的执行界面二

7.5.2　发送邮件组件

发送邮件是使用 ASP .NET 的相关组件完成的，主要涉及 MailMessage 类和 SmtpClient 类，它们的命名空间均为 System. Net. Mail。

MailMessage 类表示可以使用 SmtpClient 类发送的电子邮件。当使用 MailMessage 类创建 MailMessage 对象时，可以将电子邮件的发件人、收件人、主题和正文指定为参数。MailMessage 类的常用属性如表 7.17 所示。

表 7.17　MailMessage 类的常用属性

属　性	说　明
Body	获取或设置邮件正文
BodyEncoding	获取或设置用于邮件正文的编码
From	获取或设置此电子邮件的发信人地址
Headers	获取与此电子邮件一起传输的电子邮件标头
HeadersEncoding	获取或设置此电子邮件的用户定义的自定义标题使用的编码
Priority	获取或设置此电子邮件的优先级
ReplyToList	获取或设置邮件的回复地址的列表
Sender	获取或设置此电子邮件的发件人地址
Subject	获取或设置此电子邮件的主题行
SubjectEncoding	获取或设置此电子邮件的主题内容使用的编码
To	获取包含此电子邮件的收件人的地址集合

SmtpClient 类用于将电子邮件发送到 SMTP 服务器以便传递。SmtpClient 类的常用属性如表 7.18 所示，其常用方法如表 7.19 所示。

表 7.18　SmtpClient 类的常用属性

属　性	说　明
Credentials	获取或设置用于验证发件人身份的凭据
DeliveryFormat	获取或设置 SmtpClient 用于发送电子邮件的传递格式
DeliveryMethod	指定如何处理待发的电子邮件
Host	获取或设置用于 SMTP 事务的主机的名称或 IP 地址
Port	获取或设置用于 SMTP 事务的端口。
ServicePoint	获取用于传输电子邮件的网络连接

表 7.19　SmtpClient 类的常用方法

方　　法	说　　明
OnSendCompleted	引发 SendCompleted 事件
Send(MailMessage)	将指定的邮件发送到 SMTP 服务器以便传递
Send(String,String,String,String)	将指定的电子邮件发送到 SMTP 服务器以便传递。使用 String 对象指定邮件的发件人、收件人、主题和邮件正文
SendAsync(MailMessage,Object)	将指定的电子邮件发送到 SMTP 服务器以便传递。此方法不会阻止调用线程,并允许调用方将对象传递给操作完成时调用的方法
SendAsyncCancel	取消异步操作以发送电子邮件
SendMailAsync(MailMessage)	将指定的消息以异步操作发送给配送的 SMTP 服务器

【练一练】 在本章网站 CH7 中添加一个 webform4 网页,其功能是说明发送邮件的过程。其设计步骤如下:

① 打开 CH7 网站,选择“网站|添加新项”菜单命令,出现“添加新项-CH7”对话框,在中间列表中选择“Web 窗体”,将文件名称改为 webform4. aspx,其他保持默认项,单击“添加”按钮。

② 本网页的设计界面如图 7.12 所示,有一个 7 行 1 列的表格,表格中包含相关控件。

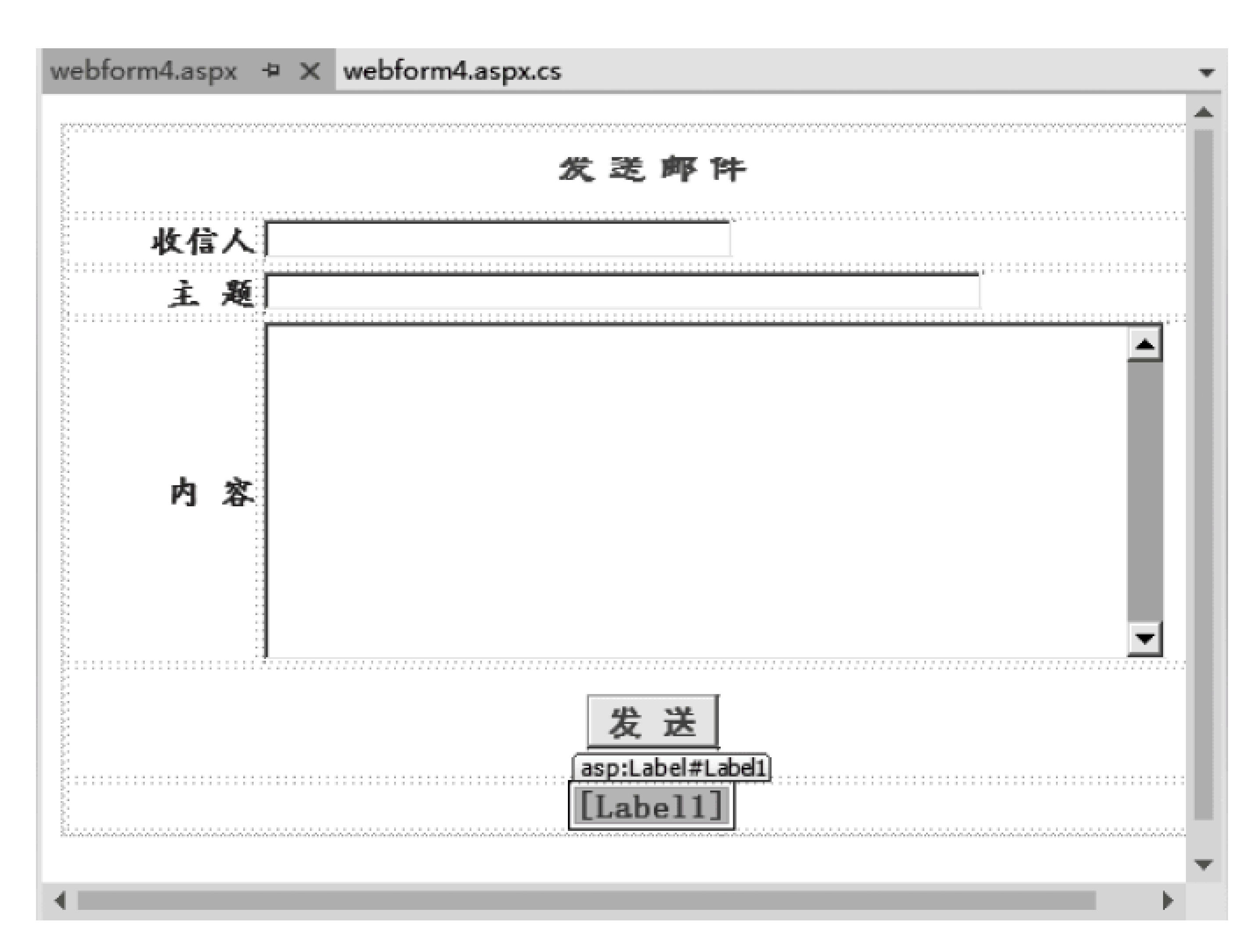


图 7.12　webform4 网页的设计界面

其源视图代码如下:

```
<%@ Page Language = "C#" AutoEventWireup = "true" CodeFile = "webform4.aspx.cs"
    Inherits = "webform4" %>
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head id = "Head1" runat = "server">
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8"/>
    <title></title>
    <link href = "../App_Themes/StyleSheet.css" rel = "stylesheet" />
    <style type = "text/css">
        .auto-style1 {
            font-family: 楷体; font-size: medium;
            color: #0000FF;     font-weight: bold;
            text-align: right; height: 22px; width: 14%;
        }
```

```
  </style>
 </head>
 <body>
  <form id = "form1" runat = "server">
   <div>
    <table id = "tablecenter" style = "width:563px">
     <tr>
      <td colspan = "2" style = "text - align: center; height: 40px;">
         <span style = "color: #FF0000; font - size: large; font - weight: 700;
             font - style: normal; font - family: 华文隶书">发 送 邮 件</span>
      </td>
     </tr>
     <tr>
      <td class = "auto - style1">收信人</td>
      <td style = "width:70 % ">
         <asp:TextBox ID = "TextBox1" runat = "server" Width = "216px" />
      </td>
     </tr>
     <tr>
      <td class = "auto - style1">主 题</td>
      <td><asp:TextBox ID = "TextBox2" runat = "server" Width = "335px" /></td>
     </tr>
     <tr>
      <td class = "auto - style1">内 容</td>
      <td>
        <asp:TextBox ID = "TextBox3" runat = "server" Height = "156px"
            TextMode = "MultiLine" Width = "425px" style = "font - size:
            medium; font - weight: 700; font - family: 仿宋" />
      </td>
     </tr>
     <tr>
      <td colspan = "2" style = "text - align:center; height: 53px">
         <asp:Button ID = "Button1" runat = "server" OnClick = "Button1_Click"
            Text = "发 送" style = "color: #FF0000; font - size: medium;
                font - weight: 700; font - family: 黑体" />
      </td>
     </tr>
     <tr>
      <td colspan = "2" style = "text - align:center; height: 23px">
        <asp:Label ID = "Label1" runat = "server" style = "color: #FF00FF;
          font - size: medium; font - weight: 700; font - family: 仿宋" />
      </td>
     </tr>
    </table>
   </div>
  </form>
 </body>
</html>
```

③ 在网页上设计如下事件处理方法：

```
using System.Net.Mail;
protected void Button1_Click(object sender, EventArgs e)
{   MailMessage mail = new MailMessage();
    mail.To.Add(new MailAddress(TextBox1.Text.Trim()));
    mail.From = new MailAddress("abc@126.com");
    mail.Subject = TextBox2.Text;
    mail.Body = TextBox3.Text.Trim();
```

```
        mail.Priority = MailPriority.High;
        SmtpClient smtpmail = new SmtpClient();
        smtpmail.DeliveryMethod = SmtpDeliveryMethod.Network;
                                                   //指定电子邮件发送方式为 Network
        smtpmail.Host = "smtp.126.com";            //指定 smtp 服务器地址
        smtpmail.Credentials = new System.Net.NetworkCredential("abc", "1234");  //认证
        try
        {   smtpmail.Send(mail);  }
        catch (Exception ex)
        {   Label1.Text = "出错提示: " + ex.Message;  }
        Label1.Text = "本邮件已成功发送!";
    }
```

其中 abc@126.com 是电子邮件发送方的邮箱，密码为 1234。需要注意的是，新近注册的 126.com 邮箱已经屏蔽了这种电子邮件发送功能，早期注册的邮箱才具有这种电子邮件发送功能。

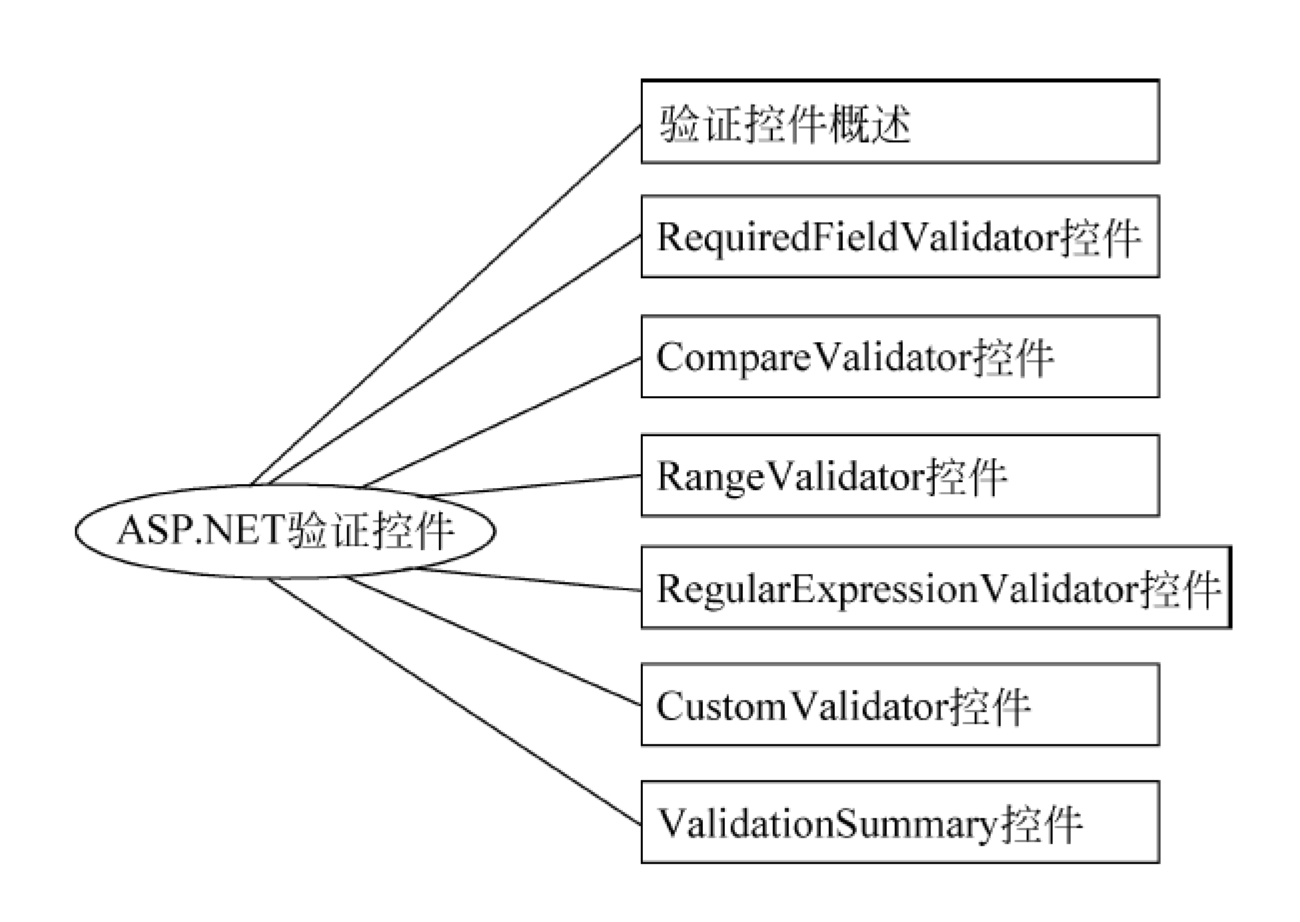
④ 单击工具栏中的 ▶ Internet Explorer 按钮运行本网页，输入发件人、主题和邮件正文，如图 7.13 所示，单击“发送”命令按钮，便可以发送邮件。



图 7.13　webform4 网页的执行界面

7.6　ASP .NET 验证控件

知识梳理



7.6.1　验证控件概述

在 ASP .NET 中提供了 6 种数据验证控件，它们位于"工具箱"的"验证"类别中，如图 7.14 所示。其中，ValidationSummary 控件(从 WebControl 类派生)用于在一个位置向用户显示所有验证错误；其余 5 个控件(都是从 BaseCompareValidator 类派生的)用来执行实际的有效性验证，如范围检查或模式匹配验证控件。

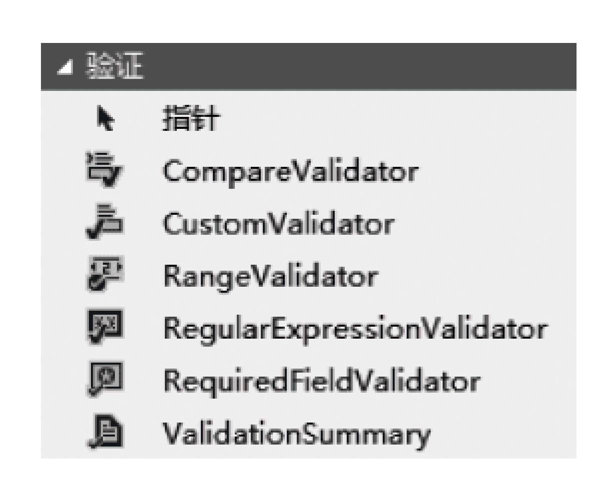


图 7.14　ASP .NET 提供的验证控件

1. 使用验证控件

通过像添加其他服务器控件那样向网页添加验证控件，即可启用对用户输入的验证。每个验证控件都引用网页上其他地方的输入控件(服务器控件)。处理用户输入时(如当提交页面时)，验证控件会对用户输入进行测试，并设置属性以指示该输入是否通过测试。调用了所有验证控件后，会在网页上设置一个属性以指示是否出现验证检查失败。

可以使用自己的代码来测试网页和单个控件的状态。例如，使用用户输入信息更新数据记录之前来测试验证控件的状态。如果检测到状态无效，将会略过更新。通常，如果任何验证检查失败，都将跳过所有处理过程并将网页返回给用户。检测到错误的验证控件随后将生成显示在网页上的错误信息。也可以将验证控件关联到验证组中，使属于同一组的验证控件可以一起进行验证。

默认情况下，在单击按钮控件(如 Button、ImageButton 或 LinkButton)时执行验证。可通过将按钮控件的 CausesValidation 属性设置为 false 来禁止在单击按钮控件时执行验证。

2. 何时进行验证

验证控件在服务器代码中执行输入检查。当用户向服务器提交网页之后，服务器将逐个调用验证控件来检查用户输入。如果在任意输入控件中检测到验证错误，则该网页将自行设置为无效状态，以便在代码运行之前测试其有效性。验证发生的时间是：已对网页进行了初始化(即处理了视图状态和回发数据)，但尚未调用任何更改或单击事件处理程序。

3. 验证多个条件

每个验证控件通常只执行一次测试，但有时需要指定多个条件。例如，指定某个日期文本框必须输入数据，同时将该用户输入限制为只接受特定范围内的日期。

此时可以将多个验证控件附加到网页上的一个输入控件。ASP .NET 使用逻辑 AND 运算符来解析控件执行的测试，这意味着用户输入的数据必须通过所有测试才能视为有效。

有些情况下，几种不同格式的输入都可能是有效的。例如，在提示输入电话号码时，可以允许用户输入本地号码、长途号码或国际长途号码。若要执行此类测试(必须仅通过一个测试的逻辑 OR 运算)，可以使用 RegularExpressionValidator 验证控件并在该控件中指定多个有效模式，或使用 CustomValidator 验证控件并编写自己的验证代码。

4. 验证控件的公共属性和方法

所有验证控件都具有服务器控件的一些常用属性，另外，5 个 ASP .NET 有效性验证控件都有一些如表 7.20 所示的公共属性。正确地设置这些公共属性是使用验证控件的关键。

表 7.20 验证控件的公共属性

属性	说　明
ControlToValidate	获取或设置要验证的输入控件的 ID
Display	获取或设置验证控件中错误信息的显示行为
EnableClientScript	指示是否启用客户端验证。通过将 EnableClientScript 属性设置为 False,可在支持此功能的浏览器上禁用客户端验证
ErrorMessage	获取或设置验证失败时 ValidationSummary 控件中显示的错误消息的文本,此属性不会将特殊字符转换为 HTML 实体。例如,小于号字符(＜)不转换为 <。这允许将 HTML 元素(如＜img＞元素)嵌入到该属性的值中
IsValid	获取或设置一个值,该值指示关联的输入控件是否通过验证
Text	获取或设置验证失败时验证控件中显示的文本
ValidationGroup	获取或设置此验证控件所属的验证组的名称

每个验证控件都对应一个需要验证其输入值的输入控件,如 TextBox 控件,验证控件的 ControlToValidate 属性就设置为这个输入控件的 ID。

如果输入控件的值通过了验证,验证控件的 IsValid 为 True,否则为 False。Page. IsValid 为 True 表示该网页的所有控件都通过了验证,否则 Page. IsValid 为 False。

使用 Display 属性指定验证控件中错误信息的显示行为,显示行为取决于是否执行客户端验证。如果客户端验证不是活动的(由于浏览器不支持它等情况),则 Static 和 Dynamic 的行为相同,即错误信息仅在显示时才占用空间,在错误信息不显示(Static)时为其动态分配空间的功能只对客户端验证适用。

当同时使用 Text 和 ErrorMessage 属性时,发生错误时将显示 Text 属性的信息。

5 个有效性验证控件都有 Validate 方法,该方法对关联的输入控件执行验证并更新 IsValid 属性。

7.6.2 RequiredFieldValidator 控件

RequiredFieldValidator 控件又称非空验证控件。该控件常用于文本框数据的非空验证,确保用户在网页上输入数据时不会跳过必选字段(必须输入数据的字段),也就是说,检查被验证控件的输入是否为空,如果为空,则在网页中显示提示信息。

除了公共属性外,RequiredFieldValidator 控件还有一个重要的属性 InitialValue,它获取或设置关联的输入控件的初始值。如果输入控件失去焦点时没有从 InitialValue 属性更改值,它将不能通过验证。

7.6.3 CompareValidator 控件

CompareValidator 控件又称比较验证控件。该控件将用户的输入与常数值(由 ValueToCompare 属性指定)、另一个控件(由 ControlToCompare 属性指定)的属性值进行比较,若不相同,则在网页中显示提示信息。

除了公共属性外,CompareValidator 控件的其他重要的属性如表 7.21 所示。

使用 Operator 属性指定要执行的比较操作,如大于、等于等,其基本取值如表 7.22 所示。如果设置 Type 属性,则在比较操作前,将两个比较值都自动转换为 Type 属性指定的数据类型,然后进行比较,Type 属性的取值有 0(String)、1(Integer)、2(Double)和 3(Date)等。

表 7.21　CompareValidator 控件的常用属性

属　性	说　明
ControlToCompare	获取或设置要与所验证的输入控件进行比较的输入控件
ValueToCompare	获取或设置一个常数值，该值要与由用户输入到所验证的输入控件中的值进行比较
Operator	获取或设置要执行的比较操作
Type	获取或设置在比较之前将所比较的值转换到的数据类型

表 7.22　Operator 属性的基本取值及其说明

操　作	值	说　明
Equal	0	默认值，所验证的输入控件的值与其他控件的值或常数值之间的相等比较
NotEqual	1	所验证的输入控件的值与其他控件的值或常数值之间的不等比较
GreaterThan	2	所验证的输入控件的值与其他控件的值或常数值之间的大于比较
GreaterThanEqual	3	所验证的输入控件的值与其他控件的值或常数值之间的大于或等于比较
LessThan	4	所验证的输入控件的值与其他控件的值或常数值之间的小于比较
LessThanEqual	5	所验证的输入控件的值与其他控件的值或常数值之间的小于或等于比较
DataTypeCheck	6	输入到所验证的输入控件的值与 BaseCompareValidator. Type 属性指定的数据类型之间的数据类型比较。如果无法将该值转换为指定的数据类型，则验证失败

通常不要同时设置 ControlToCompare 和 ValueToCompare 属性，但如果同时设置了这两个属性则 ControlToCompare 属性优先。

7.6.4　RangeValidator 控件

RangeValidator 控件又称范围验证控件。该控件确保用户输入的值在指定的上下限范围之内，当输入不在验证的范围内时，则在网页中显示提示信息。

除了公共属性外，RangeValidator 控件的其他重要的属性有 MinimumValue 和 MaximumValue，它们用于指定要比较值的范围。

7.6.5　RegularExpressionValidator 控件

RegularExpressionValidator 控件又称正则表达式验证控件，比非空验证和范围验证控件的功能更强大，对应命名空间 System. Web. UI. WebControls 中的 RegularExpressionValidator 类。

除了具有验证控件的公共属性外，RegularExpressionValidator 控件的其他重要属性有 ValidationExpression，该属性用于指定正则表达式。正则表达式由正则表达式字符组成。常用的正则表达式字符及其说明如表 7.23 所示。例如，邮政编码的正则表达式为“\d{6}”。

RegularExpressionValidator 控件确保用户输入信息与 ValidationExpression 属性指定的正则表达式模式相匹配。

表 7.23　常用正则表达式字符及其说明

正则表达式字符	说　明
[⋯]	匹配括号中的任何一个字符
[^⋯]	匹配不在括号中的任何一个字符
\w	匹配任何一个字符（a～z、A～Z 和 0～9）
\W	匹配任何一个空白字符

续表

正则表达式字符	说　明
\s	匹配任何一个非空白字符
\S	与任何非单词字符匹配
\d	匹配任何一个数字字符(0～9)
\D	匹配任何一个非数字字符(^0～9)
[\b]	匹配任何一个退格键字符
{n,m}	最少匹配前面表达式 n 次,最大为 m 次
{n,}	最少匹配前面表达式 n 次
•	匹配前面表达式 0 次或 1 次{0,1}
+	至少匹配前面表达式 1 次{1,}
*	至少匹配前面表达式 0 次{0,}
|	匹配前面表达式或后面表达式
{…)	在单元中组合项目
^	匹配字符串的开头
$	匹配字符串的结尾
\b	匹配字符边界
\B	匹配非字符边界的某个位置

实际上,Visual Studio 预先设好了一些常用的正则表达式,在“属性”窗口中单击 ValidationExpression 属性右侧的[...]按钮,出现如图 7.15 所示的“正则表达式编辑器”对话框,其中列出了常用的正则表达式,可以从中选择一个标准表达式即可。

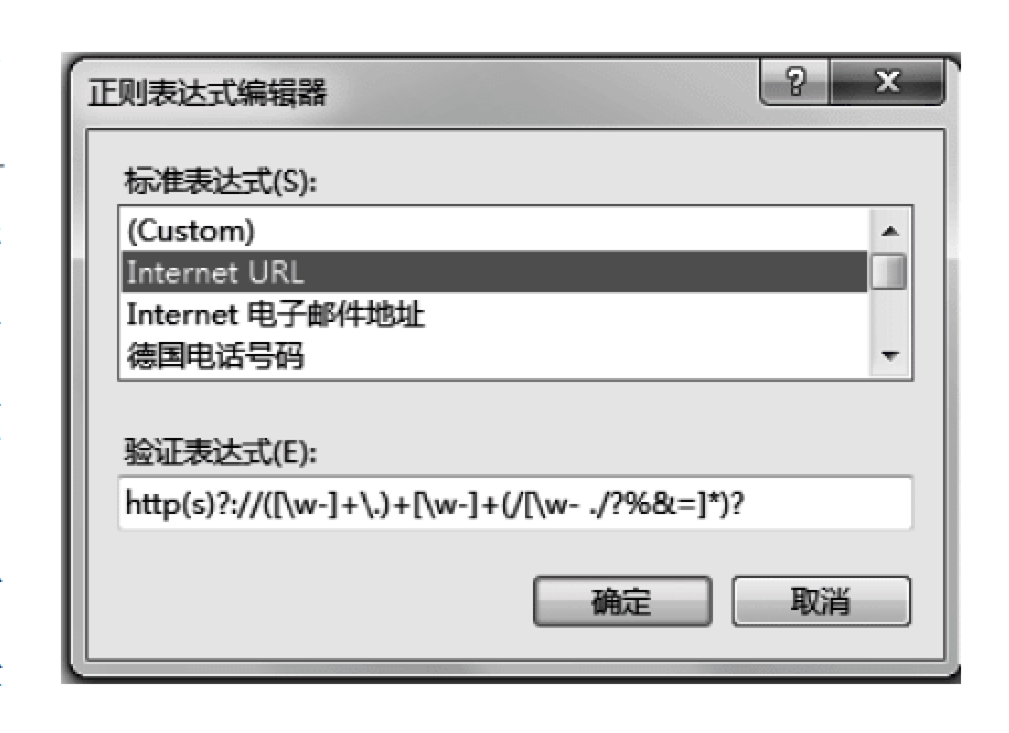


图 7.15 “正则表达式编辑器”对话框

例如,一个 RegularExpressionValidator1 用于验证文本框 TextBox1 中输入的 Internet URL 是否正确,对应的 HTML 代码如下:

```
<asp:RegularExpressionValidator ID = "RegularExpressionValidator1" runat = "server"
  ControlToValidate = "TextBox1" ErrorMessage = "URL 错误"
  ValidationExpression = "http(s)?://([\w-]+\.)+[\w-]+(/[\w- ./?%&amp;=]*)?">URL 错误
</asp:RegularExpressionValidator>
```

其中,ValidationExpression 属性是系统自动产生的,它对应 Internet URL 的正则表达式。

7.6.6 CustomValidator 控件

CustomValidator 控件又称自定义验证控件。该控件确保用户输入的内容符合自己创建的验证逻辑(验证函数)。

除了具有验证控件的公共属性外,CustomValidator 控件具有重要的自定义验证函数属性。CustomValidator 控件的验证分为服务器自定义验证和客户端自定义验证。

对于服务器自定义验证,需要定义 OnServerValidate 属性(实际上是委托),即需为执行验证的 ServerValidate 事件提供一个处理过程,该事件在服务器上执行验证时引发。ServerValidate 事件处理方法的格式如下:

```
protected void CustomValidator1_ServerValidate(object source,
    ServerValidateEventArgs args)
{
    //代码
}
```

可以通过使用作为参数传递到该事件处理方法 ServerValidateEventArgs 对象(即 args 参数)的 Value 属性来访问要验证输入控件中的字符串，然后将验证的结果存储在 ServerValidateEventArgs 对象的 IsValid 属性中。

对于客户端自定义验证，需要定义 ClientValidationFunction 属性，该属性获取或设置用于验证的自定义客户端脚本函数的名称，即为输入控件提供用户定义的验证函数。客户端验证函数的基本格式如下：

```
<script type="text/javascript">
    function valid(oSrc, args) {
        args.IsValid = 验证对象满足的条件;
    }
</script>
```

其中，oSrc 参数传递引发事件的对象的引用；args 参数传递特定于要处理的事件的对象。

为了提高网页执行效率，使用 CustomValidator 控件时尽可能采用客户端验证方式。

【练一练】 在本章网站 CH7 中添加一个 webform5 网页，其功能是说明前面 5 个验证控件的使用方法。

其设计步骤如下：

① 打开 CH7 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH7”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform5.aspx，其他保持默认项，单击“添加”按钮。

② 设计本网页，插入一个 9×3 的表格，合并第 1 行的 3 列，输入“用户注册”文字。在第 2～第 7 行的第 1 列中输入文本，如“用户名”等，在这些行的第 2 列中各添加一个文本框，分别为 TextBox1～TextBox6(TextBox2 和 TextBox3 的 TextMode 属性设为 Password)。在第 8 行添加一个命令按钮 Button1(Text 属性设为“确定”)，在第 9 行添加一个标签 Label1(Text 属性设为空)。

③ 在第 2～第 7 行的第 3 列中依次添加 RequiredFieldValidator1 控件、RequiredFieldValidator2 控件、CompareValidator1 控件、RangeValidator1 控件、RegularExpressionValidator1 控件和 CustomValidator1 控件，如图 7.16 所示。



图 7.16 webform5 网页的初始设计界面

④ 设置各验证控件的 ErrorMessage 属性，得到如图 7.17 所示的最终设计界面。

图 7.17 webform5 网页的最终设计界面

⑤ 设置各验证控件的验证属性如下：

- 设置 RequiredFieldValidator1 控件的 ControlToValidate 属性为 TextBox1。设置 RequiredFieldValidator2 控件 ControlToValidate 属性为 TextBox2。
- 设置 CompareValidator1 控件的 ControlToValidate 属性为 TextBox3，ControlToCompare 属性为 TextBox2。
- 设置 RangeValidator1 控件的 ControlToValidate 属性为 TextBox4，MaximunValue 属性为 90，MinimunValue 属性为 10，Type 属性为 Integer。
- 设置 RegularExpressionValidator1 控件的 ControlToValidate 属性为 TextBox5，ValidationExpression 属性为"Internet 电子邮件地址"。
- 设置 CustomValidator1 控件的 ControlToValidate 属性为 TextBox6，ClientValidationFunction 属性为 valid(valid 为客户端验证函数)。

⑥ 本网页的源视图代码如下(其中包含 valid 客户端验证函数的代码)：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="webform5.aspx.cs"
    Inherits="webform5" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <script type="text/javascript">
        function valid(oSrc, args) {
            args.IsValid = (args.Value.length==11);
        }
    </script>
    <style type="text/css">
        .auto-style1 {
            font-family: 宋体; font-weight: bold; font-size: 20px;
            color: #FF0000; text-align:center;  }
        .auto-style2 {
            color: #0000FF; font-size: medium; font-weight: 700;
            font-family: 楷体; text-align:right; }
        .auto-style3 {
            font-size: medium; font-family: 仿宋; }
    </style>
  </head>
```

```
<body>
  <form id="form1" runat="server">
    <div>
      <table>
          <tr>
            <td class="auto-style1" colspan="3">用 户 注 册</td>
          </tr>
          <tr>
            <td class="auto-style2">用户名</td>
            <td><asp:TextBox ID="TextBox1" runat="server" /></td>
            <td><asp:RequiredFieldValidator ID="RequiredFieldValidator1"
                   runat="server" ErrorMessage="必须输入用户名"
                   ControlToValidate="TextBox1" CssClass="auto-style3" />
            </td>
          </tr>
          <tr>
            <td class="auto-style2" >密码</td>
            <td><asp:TextBox ID="TextBox2" runat="server" TextMode="Password" />
            </td>
            <td ><asp:RequiredFieldValidator ID="RequiredFieldValidator2"
                  runat="server" ErrorMessage="必须输入密码"
                  ControlToValidate="TextBox2" CssClass="auto-style3" />
            </td>
          </tr>
          <tr>
            <td class="auto-style2" >确认密码</td>
            <td><asp:TextBox ID="TextBox3" runat="server" TextMode="Password" />
            </td>
            <td ><asp:CompareValidator ID="CompareValidator1" runat="server"
                  ErrorMessage="两次密码输入必须相同"
                  ControlToCompare="TextBox2" ControlToValidate="TextBox3"
                  CssClass="auto-style3" />
            </td>
          </tr>
          <tr>
            <td class="auto-style2" >年龄</td>
            <td><asp:TextBox ID="TextBox4" runat="server" /></td>
            <td><asp:RangeValidator ID="RangeValidator1" runat="server"
                   ErrorMessage="年龄在 10-90 之间" ControlToValidate="TextBox4"
                   CssClass="auto-style3" MaximumValue="90"
                   MinimumValue="10" Type="Integer" />
            </td>
          </tr>
          <tr>
            <td class="auto-style2" >Email</td>
            <td ><asp:TextBox ID="TextBox5" runat="server" /></td>
            <td ><asp:RegularExpressionValidator
                   ID="RegularExpressionValidator1" runat="server"
                   ErrorMessage="邮箱格式不正确" ControlToValidate="TextBox5"
                   CssClass="auto-style3"
                   ValidationExpression="\w+([-+.']\w+)*@\w+([-.]
                        \w+)*\.\w+([-.]\w+)*" />
            </td>
          </tr>
          <tr>
            <td class="auto-style2" >手机号</td>
            <td><asp:TextBox ID="TextBox6" runat="server" /></td>
            <td><asp:CustomValidator ID="CustomValidator1" runat="server"
```

```
                    ErrorMessage = "手机号不正确" ClientValidationFunction = "valid"
                    ControlToValidate = "TextBox6" CssClass = "auto - style3" />
              </td>
            </tr>
            <tr>
              <td  colspan = "3" style = "text - align:center">
                 <asp:Button ID = "Button1" runat = "server" Text = "确定"
                    style = "color: #FF0000; font - size: medium; font - weight: 700;
                      font - family: 黑体" OnClick = "Button1_Click" />
              </td>
            </tr>
            <tr>
              <td  colspan = "3">
                <asp:Label ID = "Label1" runat = "server" style = "color: #FF00FF;
                    font - size: medium; font - weight: 700; font - family: 仿宋" />
              </td>
            </tr>
          </table>
        </div>
      </form>
    </body>
</html>
```

⑦ 在网页上设计如下事件过程：

```
protected void Button1_Click(object sender, EventArgs e)
{   if (Page.IsValid)                                       //用户输入均有效
    {    Label1.Text  =  "你的输入信息如下:<br>";
         Label1.Text  +=  " 用户名:"  +
             TextBox1.Text  +  ", 密码:"  +  TextBox2.Text  +  "<br>";
         Label1.Text  +=  " 年龄:"  +  TextBox4.Text  +
             ", 邮箱:"  +  TextBox5.Text  +  "<br>";
         Label1.Text  +=  " 手机号:"  +  TextBox6.Text;
    }
    else Label1.Text  =  "输入信息错误";
}
```

⑧ 单击工具栏中的 ▶ Internet Explorer 按钮运行本网页，如果出现如图 7.18 的错误提示，其原因是因为采用 VisualStudio 2012(或 2013)开发基于 ASP .NET 4.5 的 Web 应用程序时，很多控件默认允许了 Unobtrusive ValidationMode(一种隐式的验证方式)的属性，它与 jquery 的引用相关，但并未对其进行赋值，程序员必须手动对其进行设置。

一种简单的解决方法就是将该属性设置为 None，可以在 CH7 网站的 web.config 文件中添加如下部分达到这一目的：

```
<appSettings>
    <add key = "ValidationSettings:UnobtrusiveValidationMode"   value = "None" />
</appSettings>
```

接着可以正确执行网页，如果两次密码输入不同，单击“确定”按钮，出现如图 7.19 所示的结果；如果年龄输入错误，单击“确定”按钮，出现如图 7.20 所示的结果；如果手机号输入不同只有 6 位，单击“确定”按钮，出现如图 7.21 所示的结果；如果全部输入正确，单击“确定”按钮，出现如图 7.22 所示的结果，在标签中显示用户输入的所有信息。



图 7.18　执行网页时出现的错误



图 7.19　webform5 网页的执行界面一

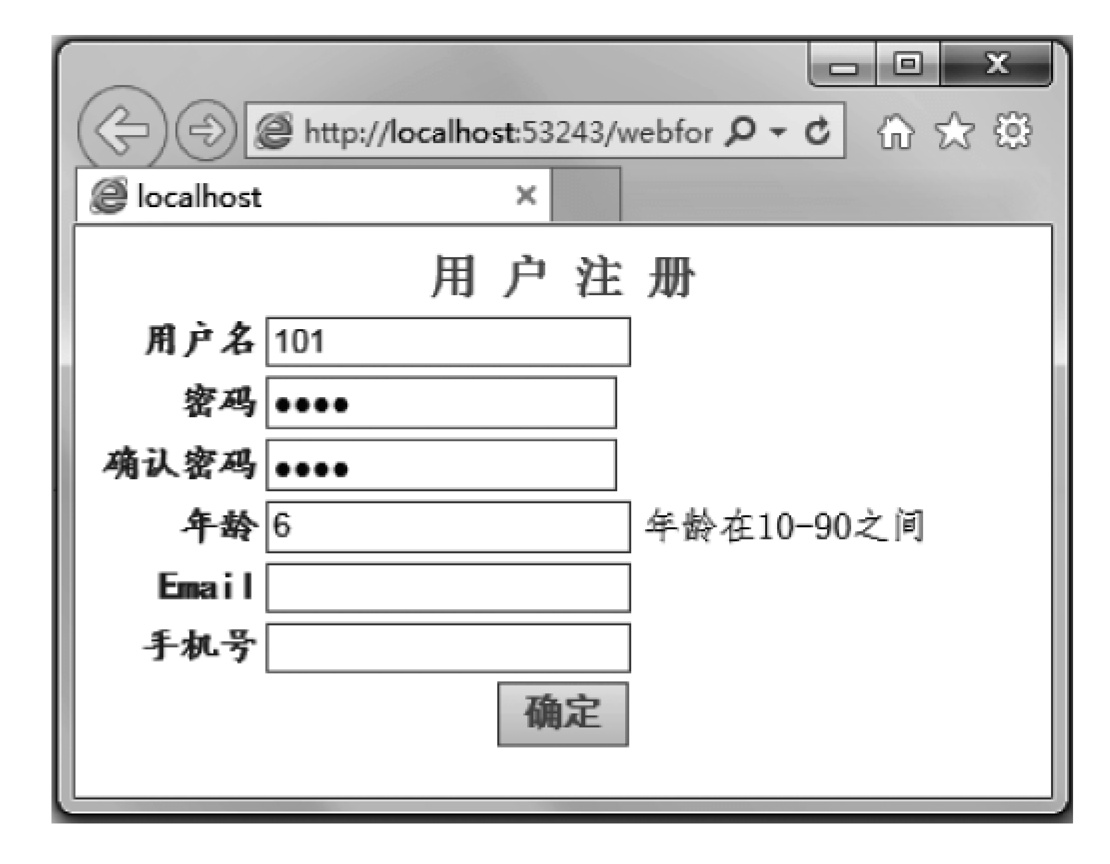


图 7.20　webform5 网页的执行界面二



图 7.21　webform5 网页的执行界面三



图 7.22　webform5 网页的执行界面四

7.6.7 ValidationSummary 控件

ValidationSummary 控件又称错误总结控件。该控件提供一个集中显示验证错误信息的地方，将本网页中所有验证控件错误信息组织好并一同显示出来。ValidationSummary 控件的常用属性如表 7.24 所示。

表 7.24 ValidationSummary 控件的常用属性

属性	说　明
HeaderText	控件汇总信息
DisplayMode	获取或设置验证摘要的显示模式
EnableClientScript	获取或设置一个值，用于指示 ValidationSummary 控件是否使用客户端脚本更新自身
ShowMessageBox	获取或设置一个值，该值指示是否在消息框中显示验证摘要。如果在消息框中显示验证摘要，则为 true，否则为 false，默认为 false
ShowSummary	获取或设置一个值，该值指示是否内联显示验证摘要

根据 DisplayMode 属性的设置，摘要可以按列表、项目符号列表或单个段落的形式显示。通过分别设置 ShowSummary 和 ShowMessageBox 属性，可在网页上和消息框中显示摘要。

注意：如果 ShowMessageBox 和 ShowSummary 属性都设置为 true，则在消息框和网页上都显示验证摘要。

7.7 练　习　题

1. 单项选择题

(1) 有关 C# 脚本代码和 HTML 说法正确的是(　　)。

A. HTML 元素的 ID 属性值直接对应 C# 脚本代码的一个对象名

B. 一个 HTML 元素不需要添加"runat=server"，C# 脚本代码也可以引用该 HTML 元素

C. 一个网页的 C# 脚本代码可以直接引用另一个网页的 HTML 元素

D. ASP .NET 在原有的 HTML 元素的基础之上，又新增加一整套服务器控件，加强页面内容和脚本的设计能力

(2) 如果希望控件内容变换后立即回传表单，需要在控件中添加属性(　　)。

A. AutoPostBack=True　　B. IsPostBack=True

C. IsPostBack=False　　D. AutoPostBack=False

(3) 当需要用控件来输入性别(男，女)或婚姻状况(已婚，未婚)时，为了简化输入，应该选用的控件是(　　)。

A. RadioButton　　B. CheckBoxList

C. CheckBox　　D. 以上都不对

(4) 当需要用控件来输入个人特长(多个)时，为了简化输入，应该选用的控件是(　　)。

A. RadioButton　　B. CheckBoxList

C. RadioButtonList　　D. 以上都不对

(5) 当需要通过班号查询某个班的学生信息时，假设班数超过 20 个，为了简化输入，班号应该选用的控件是(　　)。

A. RadioButton　　B. CheckBoxList

C. RadioButtonList　　D. DropDownList

(6) Label 服务器控件的(　　)属性用于指定控件显示的文字。

A. Width　　B. alt　　C. Text　　D. name

(7) TextBox 控件的(　　)属性值用于设置多行文本显示。

A. Text　　B. Password　　C. maxLength　　D. Multiline

(8) 要掩盖 TextBox 控件中文本，需要将控件的 TextMode 属性设置为(　　)。

A. Password　　B. MultiLine　　C. SingleLine　　D. Null

(9) 要使文本框最多输入 6 个字符，需要将该控件的(　　)属性值设置为 6。

A. MaxLength　　B. Columns　　C. Rows　　D. TabIndex

(10) 若要使 TextBox 中的文字能够看见但不能被修改，应将其(　　)属性设置为 True。

A. Locked　　B. Visible　　C. Enabled　　D. ReadOnly

(11) 网页中有一个年龄文本框 txtAge，以下(　　)代码可以获得文本框中的年龄值。

A. intage=txtAge;

B. intage=txtAge. Text;

C. intage=Convert. ToInt32(txtAge);

D. intage=int. Parse(txtAge. Text);

(12) 要使 Button 控件不可用，需要将控件的(　　)属性设置为 false。

A. EnableViewState　　B. Visible

C. CausesValidation　　D. Enabled

(13) 要使 RadioButton 控件被选中，需要将其(　　)属性设置为 true。

A. Enabled　　B. Visible　　C. Checked　　D. AutoPostBack

(14) 引用 ListBox1(列表框)最后一个数据项应使用(　　)表达式。

A. ListBox1. Items[ListBox1. Items. Count]

B. ListBox1. Items[ListBox1. SelectedIndex]

C. ListBox1. Items[ListBox1. Items. Count-1]

D. ListBox1. Items[ListBox1. SelectedIndex-1

(15) 引用 ListBox1(列表框)当前被选中的数据项应使用(　　)表达式。

A. ListBox1. Items[ListBox1. Items. Count]

B. ListBox1. Items[ListBox1. SelectedIndex]

C. ListBox1. Items[ListBox1. Items. Count-1]

D. ListBox1. Items[ListBox1. SelectedIndex-1

(16) 在设计网页时，可以通过(　　)属性向列表框控件如 ListBox1 的列表添加项。

A. Items　　B. Items. Count　　C. Text　　D. SelectedIndex

(17) DropDownList 被选中项的索引放置在(　　)属性中。

A. SelectedItem　　B. SelectedValue　　C. SelectedIndex　　D. TabIndex

(18) DropDownList 控件 Items 集合的 Count 属性值是(　　)。

A. 选择项的序号　　B. 项的总数目　　C. 选择项的数目　　D. 选择项的值

(19) DropDownList1. Items[0]. Text 值是控件的(　　)。
A. 文本　B. 选择的文本　C. 添加的文本　D. 首项的文本

(20) 语句 DropDownList1. Items[1]. Selected＝true 的作用是(　　)。
A. 使第 2 项被选中　B. 测试第 2 项是否被选中
C. 使第 2 项不能被选中　D. 使第 2 项可用

(21) 以下有关 ASP .NET 服务器验证控件的叙述中正确的是(　　)。
A. 可以在客户端直接验证用户输入,并显示出错信息
B. ASP .NET 服务器验证控件只能在服务端使用
C. 各种 ASP .NET 验证控件不具有共性,各自完成功能
D. 以上都不对

(22) RequiredFieldValidator 控件的 ErrorMessage 的属性用来(　　)。
A. 设置错误信息　B. 设置相应验证的控件
C. 定位错误类型　D. 启动错误处理程序

(23) RequiredFieldValidator 控件的 ControlToValidate 的属性用来(　　)。
A. 设置是否需要验证　B. 设置相应验证的控件
C. 设置验证方式　D. 设置验证的数据类型

(24) RangeValidator 控件用于验证数据的(　　)。
A. 类型　B. 格式　C. 范围　D. 正则表达式

(25) 要验证文本框中输入的数据是否为合法的邮编,需要使用(　　)验证控件。
A. RequiredFieldValidator　B. RangeValidator
C. CompareValidator　D. RegularExpressionValidator

(26) 如果需要确保用户输入小于 96 的值,应该使用(　　)验证控件。
A. CompareValidator　B. RangeValidtor
C. RequiredFieldValidator　D. RegularExpressionValidator

(27) 如要确保用户注册时两次密码一致,一般使用(　　)验证控件。
A. CompareValidator　B. RegularExpression
C. RequiredFieldValidator　D. RangeValidator

(28) 使用 RangeValidator 控件验证 TextBox 控件输入的数据是否为 2016 年 1 月 1 日到 2020 年 12 月 31 日,除了(　　)属性外,需要设置其他属性值。
A. MaxmumValue　B. MinimumValue
C. Type　D. Text

(29) 以下关于 CustomValidator 控件的叙述中错误的是(　　)。
A. 该控件允许用户根据程序设计需要自定义控件的验证方法
B. 该控件可以添加客户端验证方法和服务器验证方法
C. 它的 ClientValidatoFunction 属性指定客户端验证方法
D. runat 属性来指定服务器端验证方法

(30) ValidatorSummary 验证控件的作用是(　　)。
A. 检查错误总数　B. 集中显示各个验证的结果
C. 判断有无超出范围　D. 检查数值的大小

2. 问答题

(1) 什么是服务器控件？能完成什么样的功能？

(2) 简述 Web 标准服务器控件的处理过程。

(3) HTML 服务器控件和 HTML 元素的区别与联系？

(4) 如何将 HTML 控件转换为 Web 服务器控件。

(5) 如果一个网页上有一个 DropDownList1 控件，希望用户选择其中某项时立即回传至服务器，如何设计。

(6) 简述 ASP .NET 中有哪些类型的数据验证控件。

7.8　上机实验题

在 CH7 网站中添加一个 Exp 网页，其设计界面如图 7.23 所示，用户从省份下拉列表中选择一个省份后，在城市下拉列表中列出该省份所属城市，然后输入该城市的邮政编码，单击“确定”按钮在 Label1 控件显示用户输入的信息，如图 7.24 所示。

要求邮编文本框不能为空，而且邮编格式要正确。

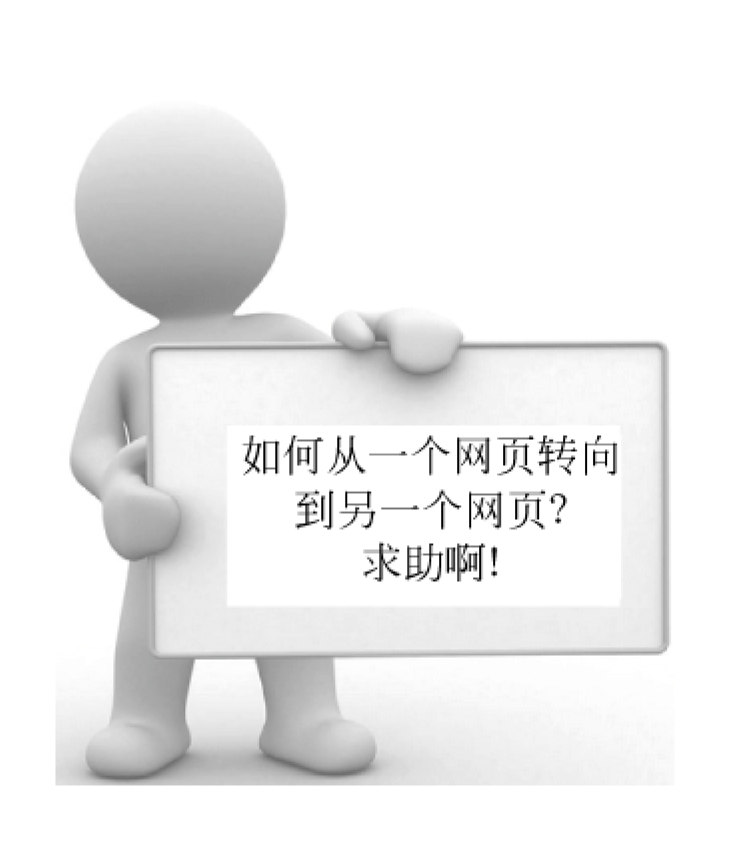


图 7.23　世界实验题网页的设计界面



图 7.24　世界实验题网页的执行界面

第 8 章　ASP .NET 内置对象



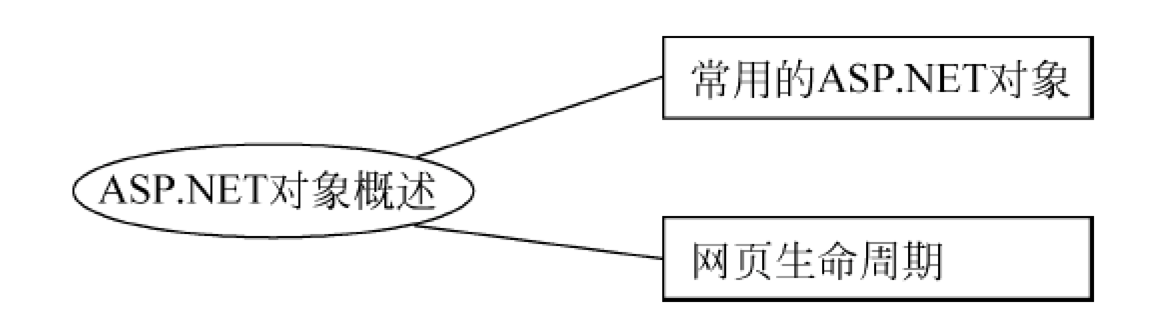


本章指南

- ASP.NET对象概述
- Page对象
- Response对象
- Request对象
- Server对象
- Application对象
- Session对象
- Cookie对象
- ViewState对象

8.1　ASP .NET 对象概述

知识梳理



8.1.1　常用的 ASP .NET 对象

在 ASP 中有几个内置对象，如 Response、Request 等，它们是 ASP 技术中最重要的一部

分。在 ASP .NET 中，这些对象仍然存在，使用的方法也大致相同，不同的是，这些内置对象是由 .NET Framework 中封装好的类来实现的。因为这些内置对象是在 ASP .NET 网页初始化请求时自动创建的，是全局变量，不需要声明就可以直接使用，如 Response. Write("Hello World")就是直接使用了 Response 对象。

ASP .NET 中常用的内置对象及其说明如表 8.1 所示，实际上，Response、Request、Server、Application 和 Session 都是 Page 类的属性。

表 8.1　ASP .NET 中常用的内置对象

对象名	说　明
Page	用于操作整个网页
Response	用于向浏览器输出信息
Request	提供对当前网页请求的访问
Server	提供服务器端的一些属性和方法
Application	提供对所有会话的应用程序范围的方法和事件的访问，还提供对可用于存储信息的应用程序范围缓存的访问
Session	用于存储特定用户的会话信息
Cookie	用于设置或获取 Cookie 信息

8.1.2　网页生命周期

ASP .NET 网页循环处理的过程如下：

① 用户通过客户端浏览器请求网页，网页第一次运行。

② Web 服务器上的 ASP .NET 对请求的网页进行处理，翻译成 HTML 和 JavaScript 等。

③ 将网页标记动态呈现到浏览器，浏览器对标记进行解析并显示。

④ 用户输入信息或从可选项中进行选择，或单击命令按钮。

⑤ 网页发送到 Web 服务器，在 ASP .NET 中称此为“回发”或“回传”。

⑥ Web 服务器执行后台代码指定的操作。

⑦ Web 服务器将执行操作后的网页以 HTML(或 XHTML)标记的形式发送到客户端浏览器。

ASP .NET 网页运行时，此网页将经历一个生命周期，在生命周期中将执行一系列处理步骤，如表 8.2 所示。

表 8.2　网页的生命周期

阶段	说　明
网页请求	网页请求发生在网页生命周期开始之前。用户请求网页时，ASP .NET 将确定是否需要分析和编译网页(从而开始网页的生命周期)，或者是否可以在不运行网页的情况下发送网页的缓存版本以进行响应
开始	在开始阶段，将设置网页属性，如 Request 和 Response。在此阶段，网页还将确定请求是回发请求还是新请求，并设置 IsPostBack 属性
网页初始化	网页初始化期间，可以使用网页中的控件，并将设置每个控件的 UniqueID 属性(服务器控件的唯一的、以分层形式限定的标识符)。此外，任何主题都将应用于网页。如果当前请求是回发请求，则回发数据尚未加载，并且控件属性值尚未还原为视图状态中的值

续表

阶段	说　明
加载	加载期间,如果当前请求是回发请求,则将使用从视图状态和控件状态恢复的信息加载控件属性
验证	在验证期间,将调用所有验证程序控件的 Validate 方法,此方法将设置各个验证程序控件和网页的 IsValid 属性
回发事件处理	如果请求是回发请求,则将调用所有事件处理程序
呈现	在呈现之前,会针对该网页和所有控件保存视图状态。在呈现阶段中,网页会针对每个控件调用 Render 方法,它会提供一个文本编写器,用于将控件的输出写入网页的 Response 属性的 OutputStream 中
卸载	完全呈现网页并已将网页发送至客户端、准备丢弃该网页后,将调用卸载。此时,将卸载网页属性(如 Response 和 Request)并执行清理

在网页命周期的每个阶段中,网页将引发开发人员编写的代码进行处理的事件,以便插入业务逻辑代码。了解页生命周期中的这些事件非常重要,因为 ASP .NET 本质上是"事件驱动"的开发模型。这样就能在合适的生命周期阶段编写代码,以达到预期效果。

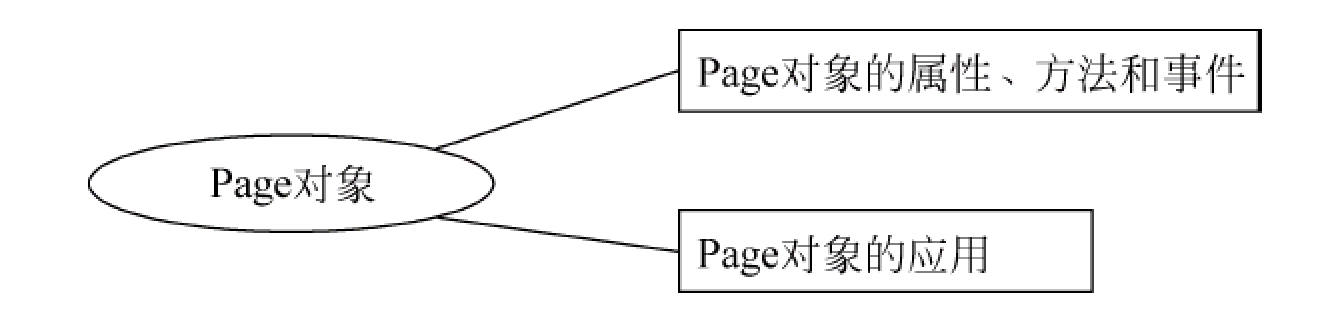
ASP .NET 中,网页常见的事件如下:

- Page_PreInit:网页初始化之前,网页上的控件还尚未创建。使用该事件来执行,检查 IsPostBack 属性来确定是不是第一次处理该网页;创建或重新创建动态控件;动态设置主题属性;读取或设置配置文件属性值。
- Page_Init:在所有控件都已初始化且已应用所有外观设置后引发。使用该事件来读取或初始化控件属性。
- Page_Load:当服务器控件加载到 Page 对象中时引发,此时网页上的控件都已生成完毕。使用该事件可以访问该事件的视图状态信息和网页 POST 数据,还可以访问网页控件层次结构内的其他服务器控件。在每次页面请求时,通知服务器控制执行 Page_Load 事件处理方法中设定的任何处理步骤。
- 控件的事件:有用户操作网页时引发,具体事件和引发的次数、顺序根据具体网页和用户的操作而定,如 Button 控件的 Click 事件或 TextBox 控件的 TextChanged 事件。在回发请求中,如果网页包含验证控件,则在执行任何处理之前检查 Page 和各个验证控件的 IsValid 属性。
- Page_PreRender:此时 HTML 网页生成完毕,接下来就要发送给客户端了,使用该事件可以在 HTML 网页上追加一些附加信息,网页页的内容进行最后更改。
- Page_Unload:执行最后的清理工作,如关闭控件特定数据库连接或完成日志记录或其他请求特定任务。

网页事件顺序是固定的,但网页上的控件事件则依赖具体网页和用户的操作。如果控件的 AutoPostBack 属性设置为 true(默认值为 false),那么将立刻引起网页回发,即网页中各个控件的值回传到 Web 服务器上,引发相应事件,由事件的处理程序处理。处理完成后如果没有跳转指令,该网页将再次发送到客户端的浏览器上。如果控件的 AutoPostBack 属性设置为 false,那么它的事件将不会一发生就马上回传,而要等到像 Button_Click 这样的回传事件发生,才一并回传。期间可能有多个事件被暂存在客户端,但是如果一个控件的一个事件重复发生,则只暂存最后一次事件的信息。

8.2　Page 对象

知识梳理



8.2.1　Page 对象的属性、方法和事件

Page 类与扩展名为 aspx 的网页文件相关联，这些文件在运行时被编译为 Page 对象，并被缓存在服务器内存中。也就是说，每一个 ASP .NET 网页都是一个 Page 对象，Page 对象是由 System. Web. UI 命名空间中的 Page 类来实现的。Page 对象充当网页中所有服务器控件的容器。

1. Page 对象的属性

Page 对象的常用属性及其说明如表 8.3 所示，此外，Page 对象还包括 Response、Request、Server、Session 和 Application 对象属性。下面介绍其中两个属性的用法。

表 8.3　Page 对象的常用属性及其说明

属　　性	说　　明
ClientQueryString	获取请求的 URL 的查询字符串部分
ErrorPage	获取或设置错误页，在发生未处理的页异常的事件时请求浏览器将被重定向到该页
Form	获取网页的 HTML 窗体
IsCrossPagePostBack	获取一个值，该值指示跨页回发中是否涉及该网页
IsPostBack	获取一个值，该值指示网页是第一次呈现还是为了响应回发而加载
IsValid	获取一个值，该值指示网页验证是否成功
Master	获取确定页的整体外观的母版页
MasterPageFile	获取或设置母版页的文件名
PreviousPage	获取向当前网页传输控件的网页

(1) IsPostBack 属性

该属性获取一个布尔值，为 True 时表示当前网页是为响应客户端回传(PostBack，指网页及操作状态传回服务器)而加载；为 False 时表示首次加载和访问网页。

该属性值是系统根据网页的执行情况自动设置的，在编程时只能读取它的值，不能修改它的值。

(2) IsValid 属性

该属性获取一个布尔值，指示网页上的验证控件是否验证成功。若网页验证控件全部验证成功，该值为 True，否则为 False。

IsValid 属性在网页验证中起着重要作用。例如，以下事件过程通过 mylabel 标签输出验证结果：

```
void Button1_Click(Object Sender, EventArgs E)
{   if (Page.IsValid)                       //也可写成 if (Page.IsValid == true)
        mylabel.Text = "信息验证成功!";
    else
        mylabel.Text = "信息验证失败";
}
```

2. Page 对象的方法

Page 对象的常用方法及其说明如表 8.4 所示。

表 8.4 Page 对象的常用方法及其说明

方　法	说　明
DataBind	将数据源绑定到被调用的服务器控件及其所有子控件
FindControl	在网页中查找指定 ID 的服务器控件
RegisterClientScriptBlock	向网页发出客户端脚本块
MapPath	检索虚拟路径(绝对的或相对的)或应用程序相关的路径映射到的物理路径
Validate	指示网页中所有验证控件进行验证

3. Page 对象的事件

Page 的常用事件及其说明如表 8.5 所示,下面对这 3 个事件作进一步的介绍。

表 8.5 Page 对象的常用事件及其说明

事件	说　明
Init	当服务器控件初始化时发生
Load	当服务器控件加载到 Page 对象中时发生
Unload	当服务器控件从内存中卸载时发生

8.2.2 Page 对象的应用

1. Page.IsPostBack 属性的基本应用

在 Page_Load 事件处理方法中,通过 Page.IsPostBack 属性可以实现首次加载和回传时执行不同的程序代码。例如:

```
void Page_Load(Object o,EventArgs e)
{   if (!Page.IsPostBack)
    {                                       //如果网页为首次加载,则进行一些操作
        ⋮
    }
    else
    {                                       //如果网页为回发,则进行一些操作
        ⋮
    }
}
```

【练一练】 以"D:\电子商务\CH8"目录创建文件系统空网站 CH8,添加一个 webform1 网页,其功能是说明 Page.IsPostBack 属性的基本应用方法。

其设计步骤如下:

① 打开 CH8 网站,选择"网站|添加新项"菜单命令,出现"添加新项-CH8"对话框,在中间列表中选择"Web 窗体",将文件名称改为 webform1. aspx,其他保持默认项,单击"添加"按钮。

② 本网页的设计界面如图 8.1 所示,其中包含一个文本框 TextBox1、一个下拉列表 DropDownList1、一个命令按钮 Button1 和两个标签(Label1 和 Label2)。将 DropDownList1 的 AutoPostBack 属性设置为 True(其默认值为 False)。在该网页上设计如下事件过程:

```
protected void Page_Load(object sender, EventArgs e)
{   if (!Page.IsPostBack)
        Label2.Text = "你尚未提交数据";
    else
    {   Label2.Text = "输入信息如下:<br>";
        Label2.Text += " 姓名为" + TextBox1.Text + ", 民族为"
          + DropDownList1.SelectedValue.ToString().Trim();
    }
}
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{
    Label1.Text = "选择" + DropDownList1.SelectedValue.ToString().Trim();
}
protected void Button1_Click(object sender, EventArgs e)
{
    //不含任何代码,起到提交网页的作用
}
```

③ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页,初始界面如图 8.2 所示,此时网页首发,Page. IsPostBack 返回 false,执行 Page_Load 事件处理方法中的 if 部分语句,在 Label2 中显示"你尚未提交数据"。



图 8.1　webform1 网页的设计界面



图 8.2　webform1 网页的执行界面一

现在文本框中输入"王华",单击"提交"按钮,其运行界面如图 8.3 所示,这是因为单击 Button1 按钮导致网页回发,Page. IsPostBack 返回 True,执行 Page_Load 事件处理方法中的 else 部分语句和 Button1_Click 事件处理方法(尽管该过程为空),在 Label2 中显示界面输入的信息。

如果在初始界面出现后,从下拉列表中选择"满族",其运行界面如图 8.4 所示,发现网页也提交了,这是怎么回事呢? 原因是 DropDownList1 控件的 AutoPostBack 属性设置为 True,当用户从中选择一个非"汉族"选项时,会导致网页提交,即回发,服务器执行 Page_Load

和 DropDownList1_SelectedIndexChanged 事件处理方法，在 Label2 中显示界面输入的信息。




图 8.3　webform1 网页的执行界面二



图 8.4　webform1 网页的执行界面三

上述网页的几点说明如下：

- 无论网页是首发还是回发，都会执行 Page_Load 事件处理方法；
- 如果 DropDownList1 控件的 AutoPostBack 属性为 False，选择其选项的操作不会导致网页提交，也不会执行 DropDownList1_SelectedIndexChanged 事件处理方法。

2. 利用 Attributes 属性实现密码回送

当一个网页含有 Password 类型的文本框时，网页提交后，该类型的文本框内容会清空，ASP .NET 这样做目的是为了安全性，但有时会给用户操作带来不便，为此可以在 Page_Load 事件处理方法中利用 Attributes 属性实现密码回送。例如，若 Password 类型的文本框为 TextBox1，可以执行如下语句达到这个目的：

```
TextBox1.Attributes.Add("value", TextBox1.Text);
```

【练一练】　在本章的 CH8 网站中添加一个 webform1-1 网页，其功能是说明利用 Attributes 属性实现密码回送的方法。

其设计步骤如下：

① 打开 CH8 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH8”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform1-1.aspx，其他保持默认项，单击“添加”按钮。

② 本网页的设计界面如图 8.5 所示，其中包含一个 4 行 2 列的表格，表格中有两个文本框（TextBox1 和 TextBox2）、一个命令按钮 Button1 和一个标签 Label1。将 TextBox2 的 TextMode 属性设置为 Password。在该网页上设计如下事件过程：



图 8.5　webform1-1 网页的设计界面

```
protected void Page_Load(object sender, EventArgs e)
{    if (!Page.IsPostBack)
        TextBox1.Attributes.Add("value", "User");
}
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = "登录信息：" + TextBox1.Text + "/" + TextBox2.Text;
}
```

Page_Load 事件处理方法的功能是在首发时将用户名文本框设置 User 的初值。

③ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，输入密码，单击"提交"按钮，其结果如图 8.6 所示，看到密码文本框内容清空了。为了保存密码文本框的内容，将 Page_Load 事件处理方法改为如下：

```
protected void Page_Load(object sender, EventArgs e)
{    if (!Page.IsPostBack)
         TextBox1.Attributes.Add("value", "User");
     else
         TextBox2.Attributes.Add("value", TextBox2.Text);
}
```

此时再执行本网页，输入密码，单击"提交"按钮，其结果如图 8.7 所示，看到密码文本框内容不变。



图 8.6　webform1-1 网页的执行界面



图 8.7　修改后 webform1-1 网页的执行界面

这是因为网页修改后，当用户单击"提交"按钮，则向服务器提交本网页，TextBox2.Text 值会传递到服务器，执行 Page_Load 的 TextBox2.Attributes.Add("value", TextBox2.Text)语句，在 ASP .NET 将 TextBox2 控件转换为客户端 HTML 标记后，在该元素上添加一个 value 属性。

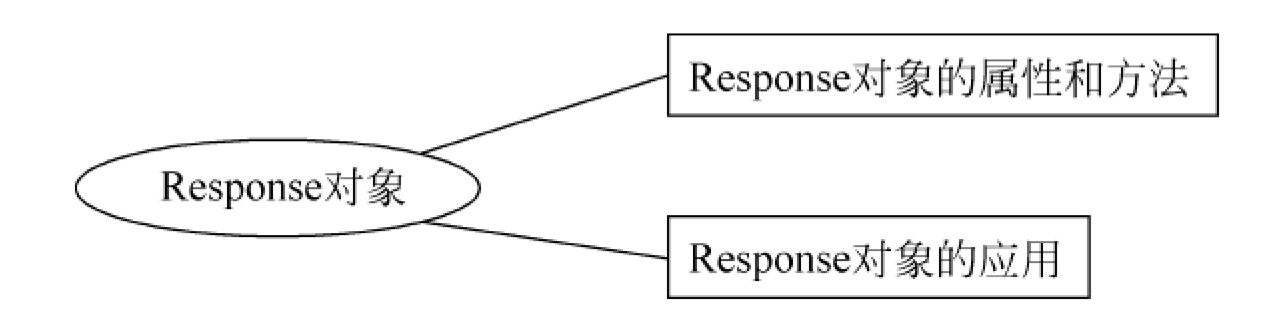
在图 8.7 中右击，在出现的快捷菜单中选择"查看源"命令，可以看到 TextBox2 的客户端 HTML 代码如下：

```
< input name = "TextBox2" type = "password" id = "TextBox2" value = "1234"
    style = "width:100px;" />
```

这就验证了本例的正确性。

8.3　Response 对象

知识梳理



8.3.1 Response 对象的属性和方法

Response 是 HttpResponse 类对象。Response 对象用于控制服务器发送给浏览器的信息，包括直接发送信息给浏览器、重定向浏览器到另一个 URL 或设置 Cookies 的值。

1. Response 对象的属性

Response 对象的常用属性及其说明如表 8.6 所示。当 Response 对象的 BufferOutput 属性为 True 时，要输出到客户端的数据都暂时存储在缓冲区内，等到所有的事件处理方法以及所有网页对象全部编译完毕后，才会将所有在缓冲区的数据发送到客户端浏览器；否则，要输出到客户端的数据直接输出，不会暂存到缓冲区。

表 8.6　Response 对象的常用属性及其说明

属　性	说　明
Buffer	获取或设置一个值，指示是否缓冲输出，并在完成处理整个响应之后将其发送
BufferOutput	获取或设置一个值，指示是否缓冲输出，并在完成处理整个网页之后将其发送
Cache	获取网页的缓存策略(过期时间、保密性、变化子句)
Cookies	获取响应 Cookies 集合
Expires	获取或设置在浏览器上缓存页过期之前的分钟数。如果用户在网页过期之前返回该页，则显示缓存版本。提供 Expires 是为了与以前版本的 ASP 兼容
IsClientConnected	获取一个值，通过该值指示客户端是否仍连接在服务器上

2. Response 对象的方法

Response 对象的常用方法及其说明如表 8.7 所示。下面介绍几个主要方法的用法。

表 8.7　Response 对象的常用方法及其说明

方　法	说　明
Output	启用到输出 HTTP 响应流的文本输出
OutputStream	启用到输出 HTTP 内容主体的二进制输出
RedirectLocation	获取或设置 HTTP“位置”标头的值
Status	设置返回到客户端的 Status 栏
AppendCookie	将一个 HTTP Cookie 添加到内部 Cookie 集合
AppendToLog	将自定义日志信息添加到 Internet 信息服务(IIS)日志文件
BinaryWrite	将一个二进制字符串写入 HTTP 输出流
Clear	清除缓冲区流中的所有 HTML 输出，不删除 Response 头信息
ClearContent	清除缓冲区流中的所有内容输出，包括删除 Response 头信息
ClearHeaders	清除缓冲区流中的所有头信息
Close	关闭到客户端的套接字连接
End	将当前所有缓冲的输出发送到客户端，停止该页的执行，并引发 EndRequest 事件
Redirect	将客户端转向到新的 URL
Write	将信息写入 HTTP 响应输出流
WriteFile	将指定的文件直接写入 HTTP 响应输出流

(1) Write 方法

使用 Write 方法可以向浏览器输出字符串。例如：

```
Response.Write("现在时间为: " + DateTime.Now.ToString());
```

用于输出当前的时间。

实际上 Write 方法将指定的字符串输出到客户端，由客户端浏览器解释后输出，所以这个输出字符串中可以包含一些 HTML 格式的输出标记。

(2) Redirect 方法

使用 Redirect 方法可以实现在不同网页之间进行转向功能，转向的网页可以是本机的网页，也可以是远程的网页。例如，输入以下代码：

```
Response.Redirect("http://www.whu.edu.cn/");
```

执行上述代码时显示的是武汉大学的主页。还可以使用 Redirect 方法在网页跳转的同时向转向的网页传递参数。例如：

```
Response.Redirect("index.aspx?a = 2&b = 10");
```

执行该语句时转向到 index. aspx 网页，并传递 a=2 和 b=10 的参数。

(3) End 方法

当执行 ASP .NET 网页时，如果遇到 End 方法，就会自动停止输出当前缓冲区的内容，并中止当前网页的处理。例如：

```
Response.Write("欢迎光临");
Response.End();
Response.Write("我的网站!");
```

只输出“欢迎光临”，而不会输出“我的网站!”。End 方法常常用来帮助调试程序。

8.3.2　Response 对象的应用

【练一练】 在本章的网站 CH8 中添加一个 webform2 网页，其功能是说明 Response 对象的使用方法。

其设计步骤如下：

① 打开 CH8 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH8”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform2. aspx，其他保持默认项，单击“添加”按钮。

② 本网页的设计界面中不包含任何控件。在该网页上设计如下事件过程：

```
protected void Page_Load(object sender, EventArgs e)
{   DateTime d = DateTime.Now;
    string mystr = "<span style='font-family: 黑体;" +
        "font-weight: bold;font-size: large;color: #FF0000;'>";
    mystr += "今日:<br>";
    mystr += " " + d.Year.ToString() + "年" +
        d.Month.ToString() + "月" + d.Day.ToString() + "日<br>";
    System.DayOfWeek week = d.DayOfWeek;
    int n = (int)week;
    if (n == 0)
        mystr += " 星期日<br>";
    else
```

```
        mystr += " 星期" + n.ToString() + "<br>";
    mystr += " 本年第" + d.DayOfYear.ToString() + "天</span>";
    Response.Write(mystr);
}
```

③ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，其执行界面如图 8.8 所示。实际上，Page_Load 事件处理方法采用 Response.Write 方法向 HTTP 响应输出流中输出如下代码：

```
<span style='font-family: 黑体;font-weight: bold;font-size: large;color: #FF0000;'>今日：<br>
   2015 年 7 月 23 日<br>
   星期 4 <br>
   本年第 204 天</span>
```

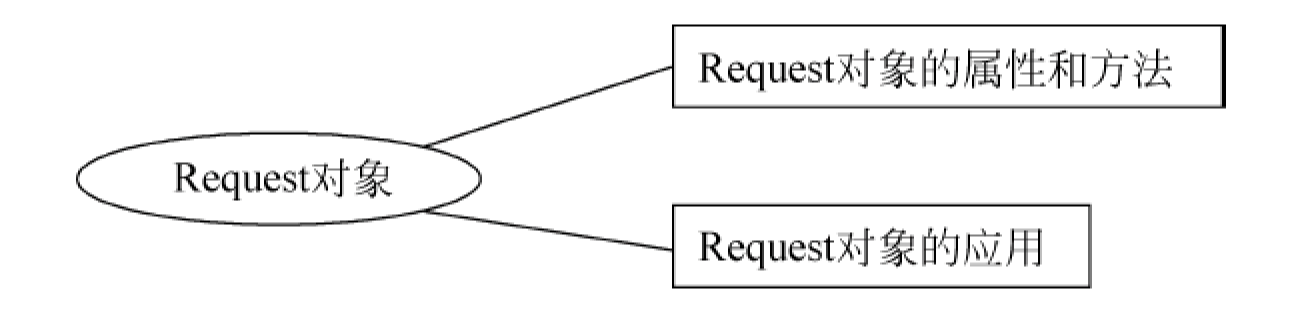
网页执行的最终结果就是上述 HTML 语句在浏览器中的呈现结果。



图 8.8　webform2 网页的执行界面

8.4　Request 对象

知识梳理



8.4.1　Request 对象的属性和方法

Request 是 HttpRequest 类对象。Request 对象的主要功能是从客户端获取数据。使用该对象可以访问任何 HTTP 请求传递的信息，包括使用 POST 方法或者 GET 方法传递的参数、Cookies 和用户验证。

1. Request 对象的属性

Request 对象的常用属性及其说明如表 8.8 所示。其中 QueryString 属性可以获取从上一个网页传递来的字符串参数，使用格式为 Request.QueryString["参数名"]。采用类似的方法可以获取 Form、Cookies 和 ServerVariable 等集合的元素值。

表 8.8　Request 对象的常用属性及其说明

属　性	说　明
ApplicationPath	获取 ASP .NET 应用的虚拟目录(URL)
PhysicalPath	获得 ASP .NET 应用的物理目录
Browser	获取有关正在请求客户的客户端的浏览器功能的信息
Cookies	获取在请求中发送的 Cookies 集
FilePath	获取当前请求的虚拟路径

续表

属　　性	说　　明
Form	获取回传到网页的窗体变量集
Headers	获取 HTTP 头部
ServerVariables	获取服务器变量的名字/值集
QueryString	获取 HTTP 查询字符串变量集合
Url	获取有关当前请求的 URL 的信息
UserHostAddress	获取客户端主机的地址

2. Request 对象的方法

Request 对象的常用方法及其说明如表 8.9 所示。下面介绍几个主要方法的用法。

表 8.9　Request 对象的常用方法及其说明

方　　法	说　　明
MapPath	返回 URL 的物理路径
SaveAs	将 HTTP 请求保存到文件中

(1) MapPath 方法

其使用语法格式如下：

```
MapPath(VirtualPath)
```

该方法将当前请求的 URL 中的虚拟路径 VirtualPath 映射到服务器上的物理路径。参数 VirtualPath 用于指定当前请求的虚拟路径(可以是绝对路径,也可以是相对路径)。返回值为与 VirtualPath 对应的服务器端物理路径。

例如,如下语句：

```
Response.Write(Request.MapPath("webform1.aspx"));
```

输出的结果是"D:\电子商务\CH8\webform1.aspx"。而如下语句：

```
Response.Write(Request.MapPath(""));
```

输出的结果是"D:\电子商务\CH8"。

(2) SaveAs 方法

其使用语法格式如下：

```
SaveAs(filename,includeHeaders)
```

该方法将客户端的 HTTP 请求保存到磁盘。参数 filename 用于指定文件在服务器上保存的位置；布尔型参数 includeHearders 用于指示是否同时保存 HTTP 头。

例如：

```
Request.SaveAs("D:\\aaa.txt", true);
```

执行后在 D 盘根目录产生 aaa.txt 文件,其中包含 Cache-Control 和 Connection 等客户端的 HTTP 请求信息。

8.4.2　Request 对象的应用

1. 获取客户端机器和浏览器的相关信息

通常使用 Request 对象的 Browser、URL、Path 和 PhysicalPath 等属性获取客户端机器和浏览器的相关信息。

【练一练】 在本章的网站 CH8 中添加一个 webform3 网页，其功能是获取客户端机器和浏览器的信息。

其设计步骤如下：

① 打开 CH8 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH8”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform3. aspx，其他保持默认项，单击“添加”按钮。

② 其设计界面中不包含任何内容。在该网页上设计如下事件过程：

```
protected void Page_Load(object sender, EventArgs e)
{   Response.Write("浏览器名称和主版本号: "
        + Request.Browser.Type + "<br>");
    Response.Write("浏览器名称: " + Request.Browser.Browser + "<br>");
    Response.Write("浏览器平台: " + Request.Browser.Platform + "<br>");
    Response.Write("客户端 IP 地址: " + Request.UserHostAddress + "<br>");
    Response.Write("当前请求的 URL: " + Request.Url + "<br>");
    Response.Write("当前请求的虚拟路径: " + Request.Path + "<br>");
    Response.Write("当前请求的物理路径: " + Request.PhysicalPath + "<br>");
}
```

③ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，其执行界面如图 8.9 所示。

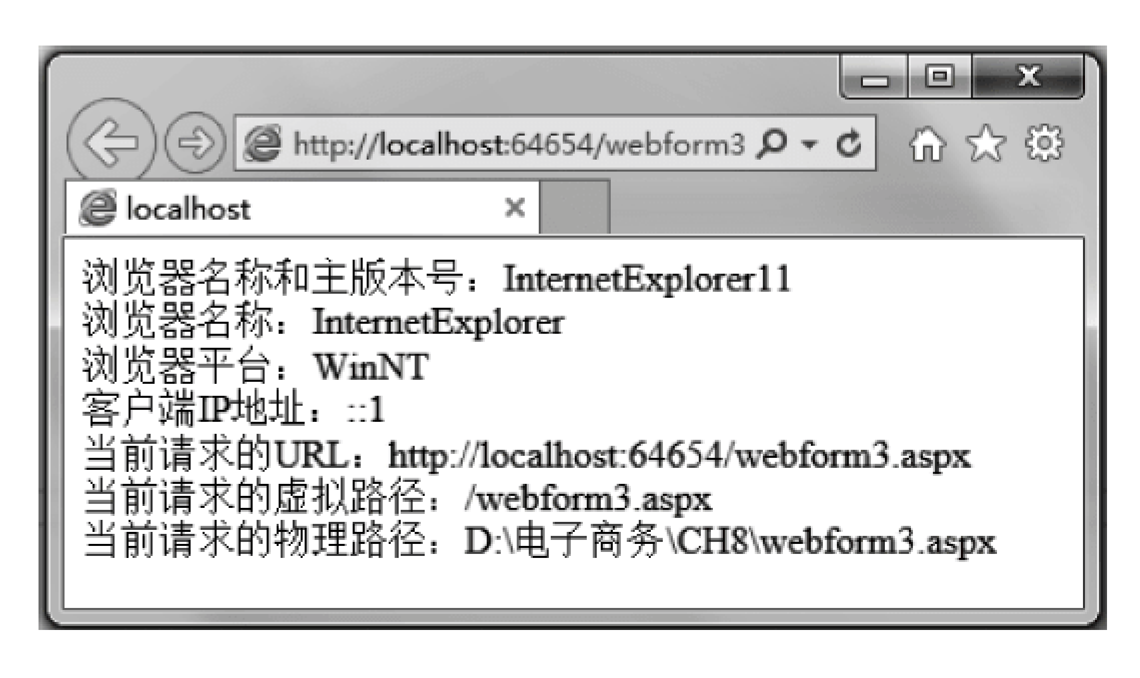


图 8.9　webform3 网页的执行界面

2. 使用 QueryString 属性在网页之间传递数据

在上网的过程中，经常发现网址后面跟一串字符，这就是通过 URL 后面的字符串在两个网页之间传递参数，QueryString 属性保存这些参数和值，因此可以通过 Request 的 QueryString 在网页之间传递信息。

【练一练】 在本章的网站 CH8 中添加 webform4 和 webform4-1 两个网页，其功能是说明 QueryString 属性的使用方法。

其设计步骤如下：

① 打开 CH8 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH8”对话框，在中

间列表中选择“Web 窗体”，将文件名称改为 webform4. aspx，其他保持默认项，单击“添加”按钮。

② webform4 网页的设计界面如图 8.10 所示，其中有两个文本框（TextBox1 和 TextBox2）和一个命令按钮 Button1。设计事件处理方法如下：

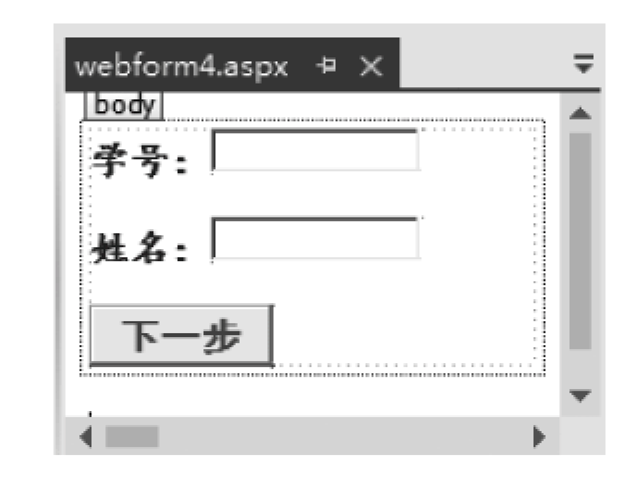


图 8.10　webform4 网页的设计界面

```
protected void Button1_Click(object sender, EventArgs e)
{    string mystr;
     mystr = "webform4 - 1.aspx?xh = " + TextBox1.Text.Trim()
          + "&xm = " + TextBox2.Text.Trim();
     Response.Redirect(mystr);
}
```

③ 再在 CH8 网站中添加一个 webform4-1 的网页，其中只有一个标签 Lable1，设计事件处理方法如下：

```
protected void Page_Load(object sender, EventArgs e)
{    string xh,xm;
     xh = Request.QueryString["xh"];
     xm = Request.QueryString["xm"];
     Label1.Text = "你的信息：<br>";
     Label1.Text += "  学号为" + xh + ",姓名为" + xm;
}
```

④ 单击工具栏中的 ▶ Internet Explorer 按钮执行 webform4 网页，输入学号和姓名，单击“下一步”按钮，出现 webform4-1 网页，如图 8.11 所示。看到显示 webform4-1 网页的地址如下：

```
http://localhost:64654/webform4 - 1.aspx?xh = 101&xm = %e7%8e%8b%e5%8d%8e
```

这就是 URL，xm 后面的数据是加密的。webform4-1 网页的 Page_Load 事件过程从中提取 QueryString 属性中对应变量的值并输出。



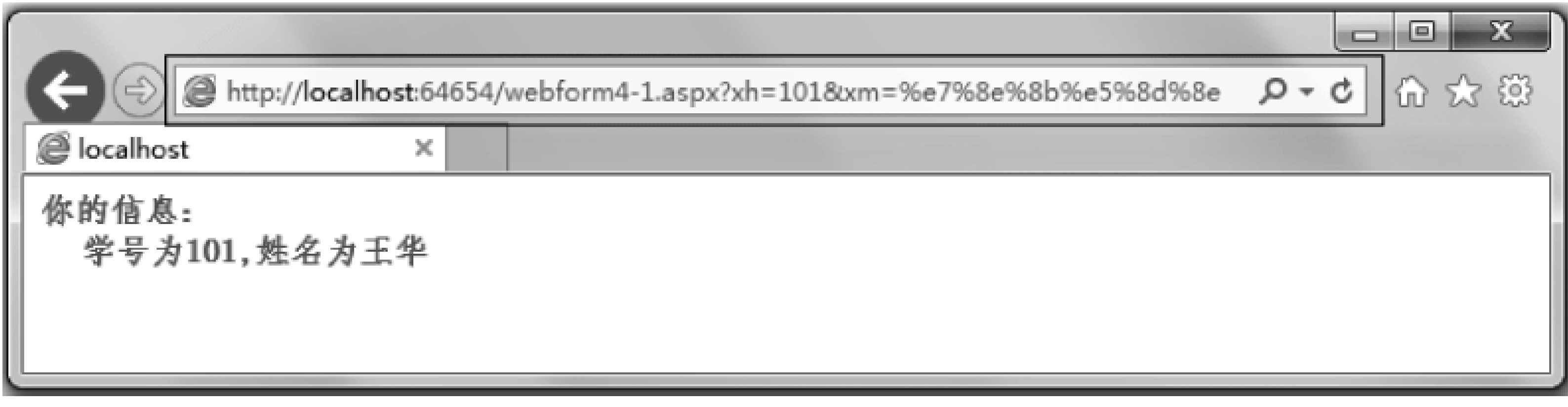


图 8.11　网页的执行过程

3. 使用 Form 属性在网页之间传递数据

使用 Request 的 Form 属性可以获取客户端通过 POST 方式传递的表单数据，从而实现网页之间的数据传递。

【练一练】 在本章的网站 CH8 中添加 webform5 和 webform5-1 两个网页，其功能是说明 Form 属性的使用方法。

其设计步骤如下：

① 打开 CH8 网站，选择"网站|添加新项"菜单命令，出现"添加新项-CH8"对话框，在中间列表中选择"Web 窗体"，将文件名称改为 webform5. aspx，其他保持默认项，单击"添加"按钮。

② webform5 网页的设计界面与 webform4 完全相同，只是 Button1 命令按钮仅仅取得提交网页的作用，其事件处理方法不含任何代码，并将 webform5 网页的<form>标记改为：

```
form id = "form1" runat = "server" method = "post" action = "webform5 - 1.aspx">
```

③ 再在 CH8 网站中添加一个 webform5-1 的网页，webform5-1 网页的设计界面与 webform4-1 完全相同，设计事件处理方法如下：

```
protected void Page_Load(object sender, EventArgs e)
{    string xh, xm;
     xh  =  Request.Form["TextBox1"];
     xm  =  Request.Form["TextBox2"];
     Label1.Text  =  "你的信息: < br >";
     Label1.Text  +=  "  学号为" + xh + ",姓名为" + xm;
}
```

④ 单击工具栏中的 ▶ Internet Explorer 按钮执行 webform5 网页，其执行结果与 webform4 完全相同。只是看到显示 webform5-1 网页的地址如下：

```
http://localhost:64654/webform5 - 1.aspx
```

其中看不到两个网页之间传递的参数值，从而达到安全的目的。webform5-1 网页的 Page_Load 事件过程提取 Form 属性中对应变量的值并输出。

上述两例都用于网页之间的数据传递，区别如下：

- webform4 例子采用 QueryString 属性传递数据，而 webform5 例子采用 Form 属性传递数据。
- webform4 例子的表单在服务器端运行，默认采用 GET 方式，而 webform5 例子的表单在客户端运行，采用 POST 方式。
- webform4 例子实现网页转向带有传递数据的参数值，而 webform5 例子通过提交命令按钮并用 action 属性指计转向的网页，不带有传递数据的参数值。
- webform4 例子通过地址直接指定传递参数值，而 webform5 例子通过提交表单，其中包含传递的参数值。

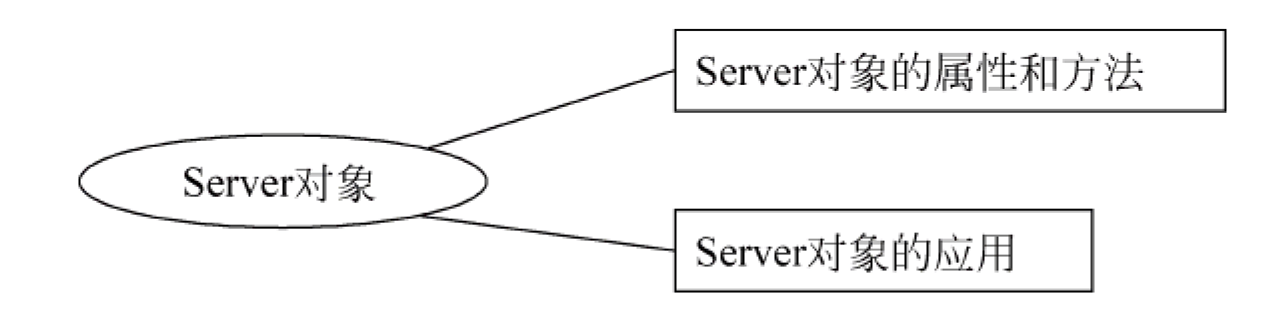
需要注意的是，webform5 例子的表单在客户端运行，webform5-1 网页中使用 Request. Form["TextBox1"]提取 TextBox1 文本框的数据。因为，TextBox1 是服务器控件，但 ASP .NET 将其转换为如下客户端代码：

```
< input name = "TextBox1" type = "text" id = "TextBox1" style = "width:80px" />
```

并且在状态视图中保存它的值，这样可以从客户端提取它的值了。

8.5　Server 对象

知识梳理



8.5.1　Server 对象的属性和方法

Server 是 HttpServerUtility 类对象。Server 对象提供了对服务器的方法和属性的访问，可以获取服务器的信息，对 HTML 文本进行编码和解码等，如文件的物理路径等。

1. Server 对象的属性

Server 对象的常用属性及其说明如表 8.10 所示。

表 8.10　Server 对象的常用属性及其说明

属　　性	说　　明
MachineName	作用是获取服务器的名称
ScriptTimeOut	获取和设置请求超时值(以秒计)

2. Server 对象的方法

Server 对象的常用方法及其说明如表 8.11 所示。下面介绍几个主要方法的用法。

表 8.11　Server 对象的常用方法及其说明

方　　法	说　　明
CreateObject	创建 COM 对象的一个服务器实例
Execute	使用另一页执行当前请求
HtmlEncode	对要在浏览器中显示的字符串进行编码
HtmlDecode	对已被编码以消除无效 HTML 字符的字符串进行解码
UrlEncode	对指定字符串以 URL 格式进行编码
UrlPathEncode	对 URL 字符串的路径部分进行 URL 编码，并返回已编码的字符串
MapPath	返回与 Web 服务器上的指定虚拟路径相对应的物理文件路径
Transfer	终止当前网页的执行，并开始执行新的请求网页

(1) MapPath 方法

使用 MapPath 方法可以获得服务器文件的物理路径。其使用语法格式如下：

```
Server.MapPath(虚拟路径字符串);
```

其用法与 Request.MapPath 类似。例如：

```
Response.Write(Server.MapPath("webform1.aspx"));
```

输出的结果是“D:\电子商务\CH8\webform1. aspx”。而如下语句：

```
Response.Write(Server.MapPath(""));
```

输出的结果是“D:\电子商务\CH8”。

(2) Transfer 方法

希望将用户从一个 ASP .NET 网页转向到另一个网页。网页转向的方法很多，使用 Server. Transfer 方法就是其中的一种方法，其语法格式如下：

```
Server.Transfer(URL);
```

Transfer 方法执行完新的网页后，不再返回原网页执行。

例如，有一个 default2. aspx 网页，它只输出“default2 网页”文字，另外一个 default1. aspx 网页，其 Page_Load 中包含以下语句：

```
Server.Transfer("default2.aspx");
Response.Write("default1 网页");
```

执行 default1. aspx 网页，当遇到该语句时转向 default2. aspx 网页，网页输出 default2. aspx 执行的结果，而不会再执行 Response. Write("default1 网页")语句，如图 8. 12 所示。

(3) Execute 方法

有时，希望在网页运行时执行其他网页的内容后继续执行当前网页的内容，可以使用 Server. Execute 方法。其语法格式如下：

```
Server.Execute(URL);
```

Execute 方法执行完新的网页后再返回原网页执行。

对于上述例子，将 default1 网页的代码中 Server. Transfer 改为 Server. Execute，其他不变。执行 default1. aspx 网页，当遇到该语句时转向到 default2. aspx 网页，网页输出 default2. aspx 执行的结果，再继续执行 default1. aspx 网页的 Response. Write("default1 网页")语句，如图 8. 13 所示。

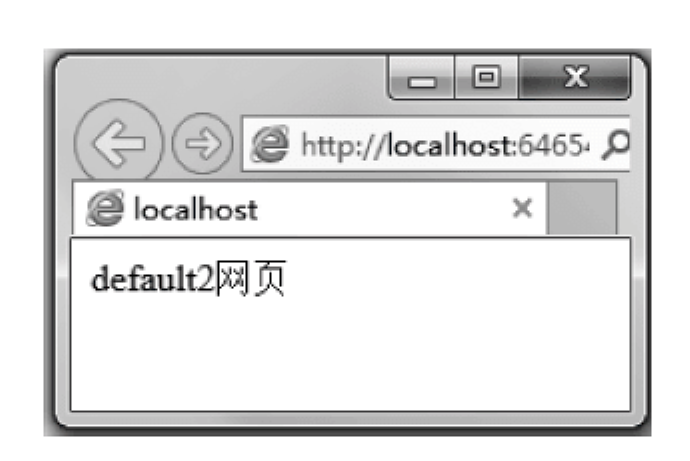


图 8. 12　使用 Server. Transfer 方法的结果

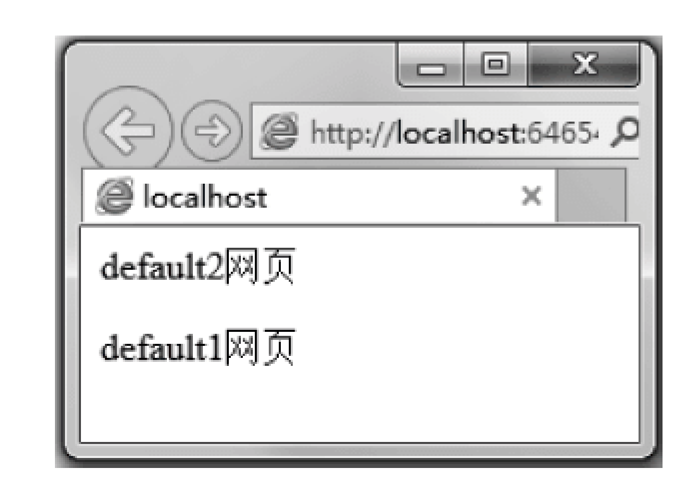


图 8. 13　使用 Server. Execute 方法的结果

8. 5. 2　Server 对象的应用

1. 使用 Server 对象获取服务器的信息

Server 对象顾名思义就是服务器对象，可以通过该对象获取服务器的信息。

【练一练】 在本章的网站 CH8 中添加一个 webform6 网页，其功能是获取服务器端相关信息。

其设计步骤如下：

① 打开 CH8 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH8”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform6. aspx，其他保持默认项，单击“添加”按钮。

② 本网页的设计界面如图 8.14 所示，其中包含两个命令按钮（Button1 和 Button2）和两个标签（Label1 和 Label2）。在该网页上设计如下事件过程：

```
protected void Button1_Click(object sender, EventArgs e)
{   Label1.Text = "服务器名称:" + Server.MachineName + "<br>" +
        "网页请求超时时间:" + Server.ScriptTimeout.ToString() + "秒";
}
protected void Button2_Click(object sender, EventArgs e)
{   string mystr1 = "<b>一个字符串</b>";
    string mystr2 = "ab12&@*%#";
    Label2.Text = "服务器路径:" + Server.MapPath(".") + "<br>" +
        "HtmlEncode:" + Server.HtmlEncode(mystr1) + "<br>" +
        "HtmlDecode:" + Server.HtmlDecode(mystr1) + "<br>" +
        "UrlEncode:" + Server.UrlEncode(mystr2) + "<br>" +
        "UrlDecode:" + Server.UrlDecode(mystr2);
}
```

③ 单击工具栏中的 ▶ Internet Explorer 按钮执行网页，分别单击其中的两个命令按钮，其结果如图 8.15 所示，从中看到 Server 对象的相关属性值和通过调用 Server 对象的相关方法返回的结果。



图 8.14　webform6 网页的设计界面



图 8.15　webform6 网页的执行界面

2. 网页转向与 Server. Transfer

Page. PreviousPage 属性在调用目标页的两种情况下都实用：在跨页回发中（这是一种基于客户端的传输）和使用 Transfer 方法时（这是基于服务器的操作）。在以上两种操作中，目标页中的代码都可以使用 PreviousPage 属性获取对源页的引用。从同一网站的源页中获取控件值的方法是，在目标页中，通过使用目标页的 PreviousPage 属性获取对源页的引用，然后调用 FindControl 方法获取对所需控件的引用。

【练一练】 在本章的网站 CH8 中添加 webform7 和 webform7-1 两个网页，其功能是说

明 Server. Transfer 的使用方法。

其设计步骤如下：

① 打开 CH8 网站，选择"网站|添加新项"菜单命令，出现"添加新项-CH8"对话框，在中间列表中选择"Web 窗体"，将文件名称改为 webform7. aspx，其他保持默认项，单击"添加"按钮。

② webform7 网页的设计界面与 webform4 完全相同，设计事件处理方法如下：

```
protected void Button1_Click(object sender, EventArgs e)
{
    Server.Transfer("webform7-1.aspx");
}
```

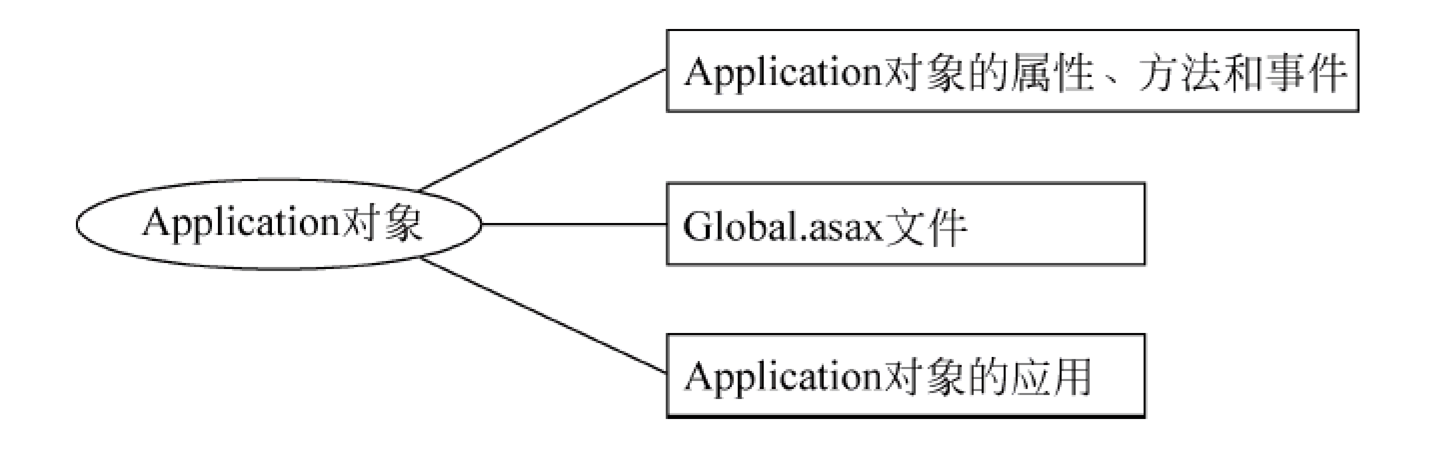
③ 再在 CH8 网站中添加一个 webform7-1 的网页，webform7-1 网页的设计界面与 webform4-1 完全相同，设计事件处理方法如下：

```
protected void Page_Load(object sender, EventArgs e)
{   string mystr = "你的信息：<br>";
    if (Page.PreviousPage != null)
    {   TextBox sBox1 = (TextBox)Page.PreviousPage.FindControl("TextBox1");
        TextBox sBox2 = (TextBox)Page.PreviousPage.FindControl("TextBox2");
        if (sBox1 != null && sBox2!= null)
            mystr += "  学号为" + sBox1.Text + ",姓名为" + sBox2.Text;
        Label1.Text = mystr;
    }
}
```

④ 单击工具栏中的 ▶ Internet Explorer 按钮执行 webform7 网页，其执行结果与 webform4 完全相同。

8.6　Application 对象

知识梳理



8.6.1　Application 对象的属性、方法和事件

Application 是 HttpApplicationState 类对象。Application 对象是运行在 Web 应用服务器上的虚拟目录及其子目录下所有文件、网页、模块和可执行代码的总和。一旦网站服务器被打开，就创建了 Application 对象；所有的用户共用一个 Application 对象并可以对其进行修改；Application 对象的这一特性使得网站设计者可以方便地创建诸如聊天室和网站计数器等常用 Web 应用程序。

Application 对象是一个对象集合,可看作是存储信息的容器,为所有用户共享。

1. Application 对象的属性

Application 对象的常用属性及其说明如表 8.12 所示。

表 8.12　Application 对象的常用属性及其说明

属性	说　明
Count	返回 Application 集合中的对象个数
Contents	表示 Application 对象中对象集合,主要是为了与以前版本的 ASP 兼容
Item	通过名称或数字索引访问对象

2. Application 对象的方法

Application 对象的常用方法及其说明如表 8.13 所示。下面介绍几个主要方法的用法。

表 8.13　Application 对象的常用方法及其说明

方法	说　明
Add	向 Application 集合中添加新对象
Clear	从 Application 集合中移除所有对象
Remove	从 Application 集合中移除指定名称的对象
RemoveAt	从 Application 集合中移除指定索引的对象
RemoveAll	从 Application 集合中移除所有对象
Lock	禁止其他用户修改 Application 集合中的对象
Unlock	允许其他用户修改 Application 集合中的对象

(1) Add 方法

用于将新对象添加到 Application 集合中。其语法格式如下:

```
Application.Add(字符串,对象值)
```

其中,“字符串”指定对象名。例如:

```
string str1 = "mystr";
int int1 = 34;
Application.Add("var1",str1);
Application.Add("var2",int1);
```

这样 Application 集合中新增了 var1 和 var2 两个对象,它们的值分别是“mystr”和 34。也可以采用以下方式新增对象:

```
Application["var1"] = str1;
Application["var2"] = int1;
```

可以采用以下方式读取 Application 集合中的信息:

```
int intvar;
string strvar;
object obj1 = Application[0];                //或 obj1 = Application.Contents[0];
objecto bj2 = Application["var2"];           //或 obj2 =  Application.Contents["var2"];
strvar = (string)obj1;                       //强制转换类型
intvar = (int)Application["var2"];           //强制转换类型
```

如果 Application 集合中指定的对象不存在，则访问该对象时返回 null。

(2) Remove 和 RemoveAt 方法

它们都用删除 Application 集合中的指定对象，其使用语法格式如下：

```
Application.Remove(对象名);
Application.RemoveAt(对象索引);
```

例如：

```
Application.Remove("var1")                    //删除 var1 对象
Application.RemoveAt(1);                      //删除 var2 对象
```

3. Application 对象的事件

Application 对象的常用事件及其说明如表 8.14 所示。这些事件处理方法应该放在 Global.asax 文件中。

表 8.14　Application 对象的常用事件及其说明

事件	说　　明
Start	在整个 ASP .NET 应用程序第一次执行时引发
End	在整个 ASP .NET 应用程序结束时引发

8.6.2　Global.asax 文件

Global.asax 文件(也称为 ASP .NET 应用程序文件)是一个可选的文件，只在希望处理应用程序事件或会话事件时，才应创建它。该文件包含响应 ASP .NET 模块所引发的应用程序级别和会话级别事件的代码。

一旦将 Global.asax 文件放在根目录或适当的虚拟目录中，ASP .NET 就会把它识别出来并且会自动使用该文件。外部用户不能下载或查看其中的代码。

例如，在 CH8 网站中创建 Global.asax 文件的过程是，打开 CH8 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH8”对话框，在中间列表中选择“全局应用程序类”，保持默认文件名，单击“添加”按钮。Global.asax 文件默认的初始代码如下：

```
<%@ Application Language="C#" %>
<script runat="server">
    void Application_Start(object sender, EventArgs e)
    {
        // 在应用程序启动时运行的代码
    }
    void Application_End(object sender, EventArgs e)
    {
        // 在应用程序关闭时运行的代码
    }
    void Application_Error(object sender, EventArgs e)
    {
        // 在出现未处理的错误时运行的代码
    }
    void Session_Start(object sender, EventArgs e)
    {
```

```
        // 在新会话启动时运行的代码
    }
    void Session_End(object sender, EventArgs e)
    {
        // 在会话结束时运行的代码.
        // 注意：只有在 Web.config 文件中的 sessionstate 模式设置为
        // InProc 时,才会引发 Session_End 事件.如果会话模式设置为 StateServer
        // 或 SQLServer,则不引发该事件.
    }
</script>
```

开发人员可以在相应的事件处理方法中添加自己的代码。

8.6.3　Application 对象的应用

【练一练】　在本章的网站 CH8 中添加一个 webform8 的网页,其功能是实现一个简单聊天室说明 Application 对象的使用方法。

其设计步骤如下:

① 打开 CH8 网站,选择“网站|添加新项”菜单命令,出现“添加新项-CH8”对话框,在中间列表中选择“Web 窗体”,将文件名称改为 webform8. aspx,其他保持默认项,单击“添加”按钮。

② 本网页的设计界面如图 8.16 所示,其中包含一个标签(Label1,用于显示聊天内容)、两个文本框(TextBox1 和 TextBox2,分别用于输入姓名和聊天记录,TextBox2 的 TextMode 属性设为 MultiLine)和一个命令按钮(Button1,用于提交聊天记录)。在该网页上设计如下事件过程:

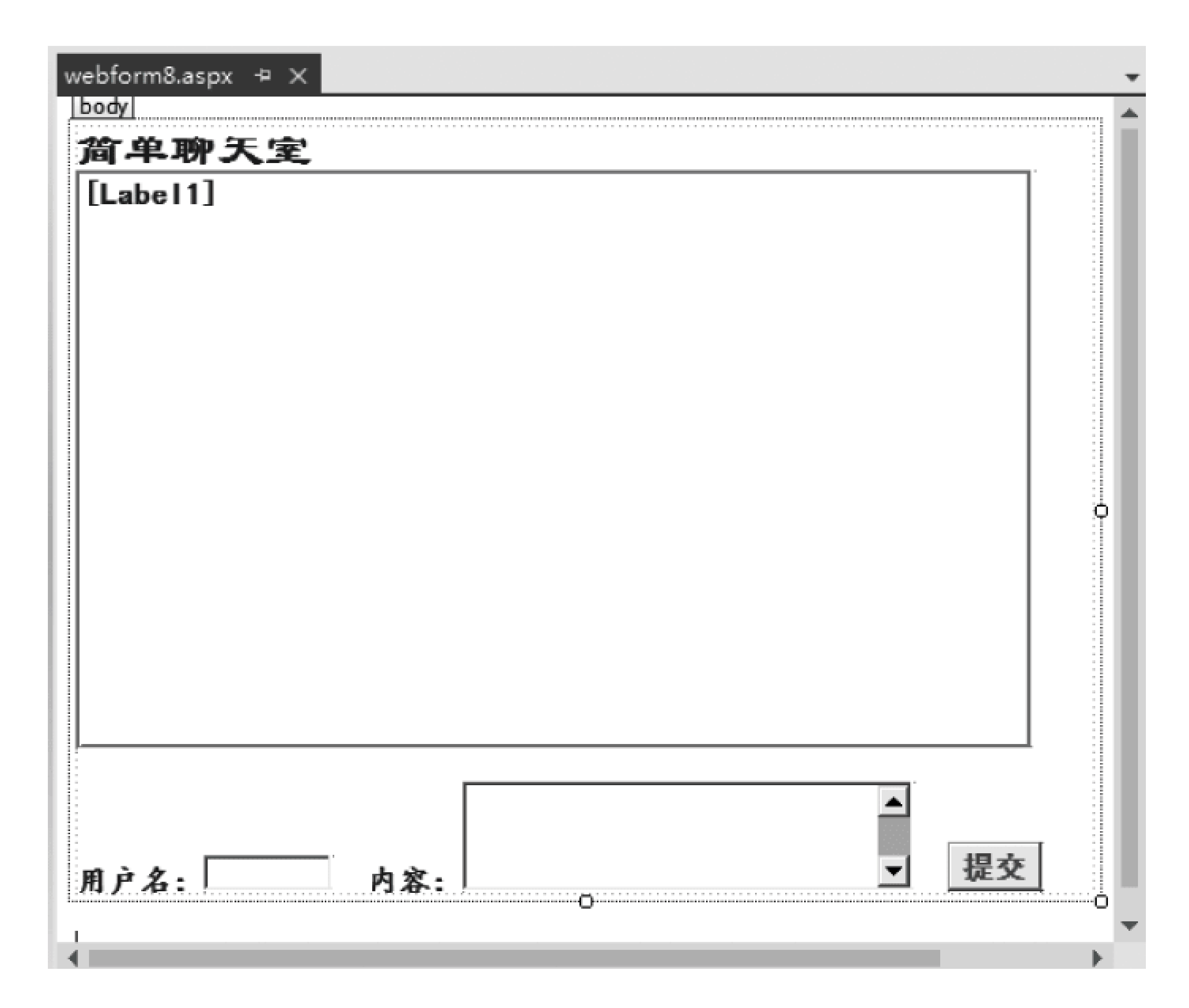


图 8.16　webform8 网页的设计界面

```
protected void Page_Load(object sender, EventArgs e)
{    if (Application["chat"]!= null)
```

```
        Label1.Text = (string)Application["chat"];
}
protected void Button1_Click(object sender, EventArgs e)
{   if (TextBox1.Text != "")
    {   Application.Lock();
        Application["chat"] = TextBox1.Text + "说:" +
            TextBox2.Text + "<br>" + Application["chat"];
        Application.UnLock();
        Label1.Text = (string)Application["chat"];
    }
    else
        Response.Write("<script>alert('必须输入用户名')</script>");
}
```

③ 创建一个 Global.asax 文件，其中包含的代码如下：

```
<%@ Application Language="C#" %>
<script runat="server">
    protected void Application_Start(object sender, EventArgs e)
    {
        Application["chat"] = null;          //聊天记录置空
    }
    protected void Application_End(object sender, EventArgs e)
    {
        Application["chat"] = null;          //聊天记录置空
    }
</script>
```

④ 单击工具栏中的 ▶ Internet Explorer 按钮执行网页，输入用户名开始聊天。启动百度浏览器，输入地址 http://localhost:64654/webform8.aspx 启动本网页，这样两个人就可以相互聊天了，如图 8.17 所示。



图 8.17　webform8 网页的执行界面

本网页设计的说明如下：

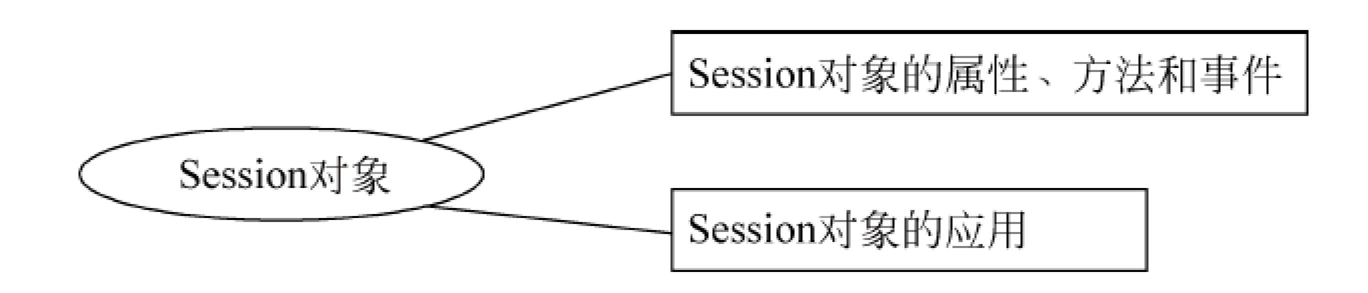
① 采用 Application["chat"]存放聊天记录(网页中当前显示最近的聊天记录)。

② Global.asax 文件中的 Application_Start 和 Application_End 两个事件过程用于在应用程序启动和退出时执行 Application ["chat"]=null 等将聊天记录清空。

③ 本网页不能即时显示另一个网友的聊天信息，只有在单击“提交”按钮后才显示所有网友的即时聊天信息。可以使用 AJAX 控件进行改进，这里不再介绍。

8.7　Session 对象

知识梳理



8.7.1　Session 对象的属性、方法和事件

Session 是 HttpSessionState 类对象。当用户请求一个 ASP .NET 网页时，系统将自动创建一个 Session(会话)，退出应用程序或关闭服务器时该会话撤销。系统在创建会话时将为其分配一个长长的字符串(SessionID)标识，以实现对会话进行管理和跟踪。该字符串中只包含 URL 中所允许的 ASCII 字符。SessionID 具有的随机性和唯一性保证了会话不会冲突，也不会被怀有恶意的人利用新 SessionID 推算出现有会话的 SessionID。

和 Application 对象一样，Session 对象也是一个对象集合，但 Session 是针对某个特定用户的，用户之间不会产生共享情况。

Application 对象和 Session 对象的区别是，Application 对象中保存的信息是为所有来访的客户端浏览器共享的，而 Session 对象保存的数据是仅为特定的来访者使用的。例如，在北京的用户 A 和上海的用户 B 同时访问某一服务器，若用户 A 修改了 Application 对象中存放的信息，用户 B 在刷新网页后就可以看到修改后的内容；但若用户 A 修改了 Session 对象中存放的数据，用户 B 是感觉不到的，此时只有用户 A 可以看到和使用这些数据，也就是说，Session 对象中存放的是专用信息。

1. Session 对象的属性

Session 对象的常用属性及其说明如表 8.15 所示。

表 8.15　Session 对象的常用属性及其说明

属　性	说　明
Count	获取会话状态集合中的项数
Item[Int32]	按数字索引获取或设置会话值
Item[String]	按名称获取或设置会话值
Keys	获取存储在会话状态集合中所有值的键的集合
SessionID	用来标识一个 Session 对象
TimeOut	获取并设置会话状态提供程序终止会话之前各请求之间所允许的超时期限(以分钟为单位)

2. Session 对象的方法

Session 对象的常用方法及其说明如表 8.16 所示。下面介绍几个主要方法的用法。

表 8.16 Session 对象的常用方法及其说明

方法	说　明
Add	将新的项添加到 Session 集合中
Clear	从 Session 集合中清除所有对象，但不结束会话
Abandon	强行结束用户会话，并清除会话中所有信息
CopyTo	将 Session 集合复制到一维数组中
Remove	删除会话状态集合中的项
RemoveAll	从会话状态集合中移除所有的键和值
RemoveAt	删除会话状态集合中指定索引处的项

(1) Add 方法

Add 方法用于将新对象添加到 Session 集合中。其语法格式如下：

```
Session.Add(字符串,对象值)
```

其中，“字符串”指定对象名。例如：

```
string str1 = "mystr";
int int1 = 34;
Session.Add("var1",str1);
Session.Add("var2",int1);
```

这样 Session 集合中新增了 var1 和 var2 两个对象，它们的值分别是“mystr”和 34。也可以采用以下方式新增对象：

```
Session["var1"] = str1;
Session["var2"] = int1;
```

可以采用以下方式读取 Session 集合中的信息：

```
int intvar;
string strvar;
object obj1 = Session[0];
objecto bj2 = Session["var2"];
strvar = (string)obj1;                        //强制转换类型
intvar = (int)Session["var2"];                //强制转换类型
```

(2) Clear 方法

Clear 方法用于清除 Session 集合中所有对象，其使用语法格式如下：

```
Session.Clear();
```

(3) Session 对象的事件

Session 对象的常用事件及其说明如表 8.17 所示。

表 8.17　Session 对象的常用事件及其说明

事件	说　　明
Start	建立 Session 对象时发生
End	结束 Session 对象时发生

8.7.2　Session 对象的应用

【练一练】 在本章的网站 CH8 中添加 webform9 和 webform9-1 两个网页，其功能是说明 Session 对象的使用方法。

其设计步骤如下：

① 打开 CH8 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH8”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform9. aspx，其他保持默认项，单击“添加”按钮。

② webform9 网页的设计界面与 webform4 完全相同，设计事件处理方法如下：

```
protected void Button1_Click(object sender, EventArgs e)
{   Session["xh"] = TextBox1.Text;
    Session["xm"] = TextBox2.Text;
    Server.Transfer("WebForm9 - 1.aspx");
}
```

③ 再在 CH8 网站中添加一个 webform9-1 的网页，webform9-1 网页的设计界面与 webform4-1 完全相同，设计事件处理方法如下：

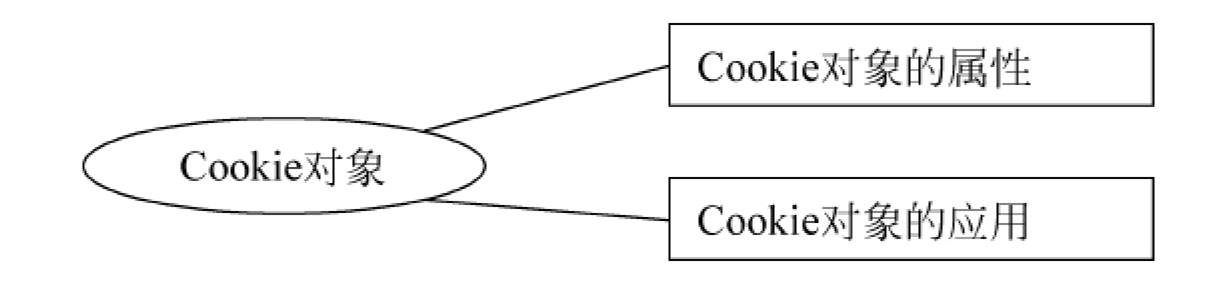
```
protected void Page_Load(object sender, EventArgs e)
{   string xh, xm;
    xh = Session["xh"].ToString();
    xm = Session["xm"].ToString();
    Label1.Text = "你的信息：<br>";
    Label1.Text += "  学号为" + xh + ",姓名为" + xm;
}
```

④ 单击工具栏中的 ▶ Internet Explorer 按钮执行 webform9 网页，其执行结果与 webform4 完全相同。

通过本例看到，Session 对象也可以用于网页之间数据传递。因此，在网页转向时数据传递的方法很多，开发人员需要根据实际情况选择合适的设计方法。

8.8　Cookie 对象

知识梳理



8.8.1　Cookie对象的属性

Response和Request对象都有一个Cookies属性，它是存放Cookie对象的集合。一个Cookie是一段文本信息，能随着用户请求和网页在Web服务器和浏览器之间传递。用户每次访问站点时，Web应用程序都可以读取Cookie包含的信息，从而知道用户上次登录的时间等具体信息。

Cookie对象和Application、Session对象一样，都是为了保存信息。它们之间的区别是，Cookie对象的信息保存在客户端，而Application和Session对象的信息保存在服务器端。

通常使用Response对象的Cookies集合属性设置Cookie信息，使用Request对象的Cookies集合属性读取Cookie信息。Cookies集合属性有Count（返回集合中Cookie对象个数）属性和Add（向Cookies集合中新增一个Cookie对象）、Clear（删除Cookies集合中所有对象）及Remove（删除Cookies集合中指定名称的对象）等方法。

Cookie对象的常用属性及其说明如表8.18所示。下面介绍其中几个主要的属性。

表8.18　Cookie对象的常用属性及其说明

属　　性	说　　明
Name	获取或设置Cookie的名称
Expires	获取或设置Cookie的过期日期和时间
Domain	获取或设置Cookie关联的域
Path	获取或设置要与Cookie一起传输的虚拟路径
Secure	获取或设置一个值，通过该值指示是否安全传输Cookie
Value	获取或设置单个Cookie值
Values	获取在单个Cookie对象中包含的键值对的集合

（1）Name属性

通过Cookie的Name属性来指定Cookie的名称，因为Cookie是按名称保存的，如果设置了两个名称相同的Cookie，后保存的那一个将覆盖前一个，所以创建多个Cookie时，每个Cookie都必须具有唯一的名称，以便日后读取时识别。

例如，mycookie是一个Cookie对象，则mycookie.Name返回该Cookie对象的名称。

（2）Value属性

Cookie的Value属性用来指定Cookie中保存的值，因为Cookie中的值都是以字符串的形式保存的，所以为Value指定值时，如果不是字符串类型要进行类型转换。

例如，mycookie是一个Cookie对象，则mycookie.Value返回该Cookie对象的值。

8.8.2　Cookie对象的应用

1. 创建Cookie对象

在.NET框架中，Cookie对象是由HttpCookie类来实现的，创建一个Cookie对象就是建立HttpCookie类的一个实例。HttpCookie类具有以下构造函数：

```
public HttpCookie (string name)
public HttpCookie(string name,string value)
```

其中，name表示Cookie对象的名称（对应Name属性），value表示Cookie对象的值（对应Value属性）。例如：

```
HttpCookie cookie1 = new HttpCookie("mycookie1");
```

```
                                            //新建名称为 mycookie1 的 Cookie 对象
cookie1.Value = "mystring";                 //其值为设为"mystring"
Response.Cookies.Add(cookie1);              //添加 cookie1 对象
HttpCookie cookie2 = new HttpCookie("mycookie2","good");
                                            //新建名称为 mycookie2 的 Cookie 对象,其值为"good"
Response.Cookies.Add(cookie2);              //添加 cookie2 对象
```

2. 设置多值 Cookie

一个 Cookie 对象可以有多个值,通过子键区分。

例如,当一个名称为 mycookie 的 Cookie 对象已添加到 Response 对象中后,可以通过以下语句设置两个子键的值:

```
Response.Cookies["mycookie"]["uname"] = "Smith";
Response.Cookies["mycookie"]["uage"] = 23.ToString();
```

或在创建 Cookie 对象同时设置多个值:

```
HttpCookie cookie = new HttpCookie("mycookie");
cookie.Values["uname"] = "Smith";
cookie.Values["uage"] = 23.ToString();
Response.Cookies.Add(cookie);
```

3. 读取 Cookie 对象

对于单值 Cookie 对象,直接用 Request.Cookies[Cookie 的 Name 属性值]来读取其 Cookie 值。

对于多值 Cookie 对象,还需加上子键名称。例如,以下语句将 Name 为 mycookie1 的 Cookie 对象的两个子键值分别在两个文本框中输出:

```
TextBox1.Text = Request.Cookies["mycookie1"]["uname"];
TextBox2.Text = Request.Cookies["mycookie1"]["uage"];
```

4. Cookie 的有效期

Cookie 的 Expires 属性为 DateTime 类型的,用来指定 Cookie 的过期日期和时间即 Cookie 的有效期。浏览器在适当时删除已经过期的 Cookie。如果不给 Cookie 指定过期日期和时间,则为会话 Cookie,不会存入用户的硬盘,在浏览器关闭后就被删除。

应根据应用程序的需要来设置 Cookie 的有效期,如果用来保存用户的首选项,则可以把其设置为永远有效(如 100 年),如果用来统计用户访问次数,则可以把有效期设置为半年。即使设置长期有效,用户也可以自行决定将其全部删除。

5. 修改和删除 Cookie

修改某个 Cookie 实际上是指用新的值创建新的 Cookie,并把该 Cookie 发送到浏览器,覆盖客户机上旧的 Cookie。

删除 Cookie 是修改 Cookie 的一种形式。由于 Cookie 位于用户的计算机中,所以无法直接将其删除。但是,可以修改 Cookie 将其有效期设置为过去的某个日期,从而让浏览器删除这个已过期的 Cookie。

【练一练】 在本章的网站 CH8 中添加一个 webform10 网页,其功能是说明 Cookie 对象的使用方法。

其设计步骤如下:

① 打开 CH8 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH8”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform10.aspx，其他保持默认项，单击“添加”按钮。

② 本网页的设计界面如图 8.18 所示，其中包含两个文本框（TextBox1 和 TextBox2）和两个命令按钮（Button1 和 Button2）。在该网页上设计如下事件过程：



图 8.18　webform10 网页的设计界面

```
protected void Button1_Click(object sender, EventArgs e)
{   //写入 Cookie 事件过程
    HttpCookie mycookie = new HttpCookie("cookie");
    mycookie.Value = TextBox1.Text;
    mycookie.Expires = DateTime.Now.AddMinutes(1);     //保存 1 分钟
    Response.Cookies.Add(mycookie);
}
protected void Button2_Click(object sender, EventArgs e)
{   //读取 Cookie 事件过程
    if (Request.Cookies["cookie"] != null)
        TextBox2.Text = Request.Cookies["cookie"].Value;
    else
        Response.Write("<script>alert('Cookie 已过期')</script>");
}
```

③ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，运行本网页，在 TextBox1 文本框中输入一个字符串，单击“写入 Cookie”按钮，然后立即单击“读取 Cookie”按钮，则在 TextBox2 文本框输出 Cookie 的值，如图 8.19 所示。如果过了 1 分钟，再单击“读取 Cookie”命令按钮，由于 Cookie 已过期，弹出如图 8.20 所示的警告框。



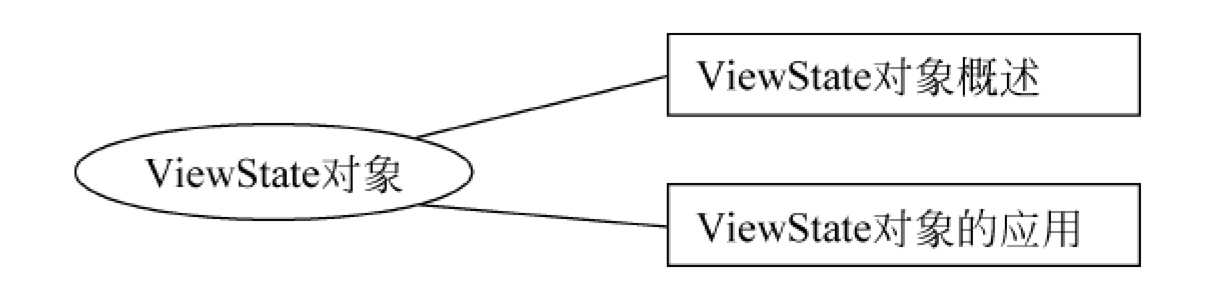
图 8.19　webform10 网页的执行界面一



图 8.20　webform10 网页的执行界面二

8.9　ViewState 对象

知识梳理



8.9.1　ViewState 对象概述

访问网站遵循 HTTP 协议，即浏览器和服务器之间通信都遵守 HTTP 协议，而 HTTP 是无状态的。其无状态的根本原因是，浏览器和服务器使用 Socket 通信，服务器将请求结果返回给浏览器后，会关闭当前 Socket 连接，而且服务器会在处理页面完毕后销毁页面对象。

这样对网站造成的影响是，如果用户输入了一些信息，当跳转到下一个页面时，数据丢失，再也不能获得那些数据。如果要知道上一次的状态信息，就得把这个状态信息记录在某个地方，包括服务器端 Session 对象、浏览器端 Cookie 对象或表单中的隐藏域等。

ASP .NET 中的状态(信息)保存方案如图 8.21 所示。其中，4 个重要的对象如下：

- ViewState(视图状态)对象：它是 ASP .NET 的 aspx 网页特有、属于网页级的，就是在页面上的一个隐藏域中保存客户端单独使用的数据的一种方式，服务器端控件的值都自动保存在 ViewState 中。
- Cookie 对象：HTTP 协议下的一种方式，通过该方式服务器或脚本能够在客户机上维护状态信息，就是在客户端保存客户端单独使用的数据的一种方式。
- Session 对象：在服务器端保存客户端单独使用的数据的一种方式。就像银行账户，钱都存在银行里，就拿一张银行卡"所谓的 SessionId"回家(写入客户端的 Cookie 中)。
- Application 对象：在服务器端保存共享数据的一种方式。

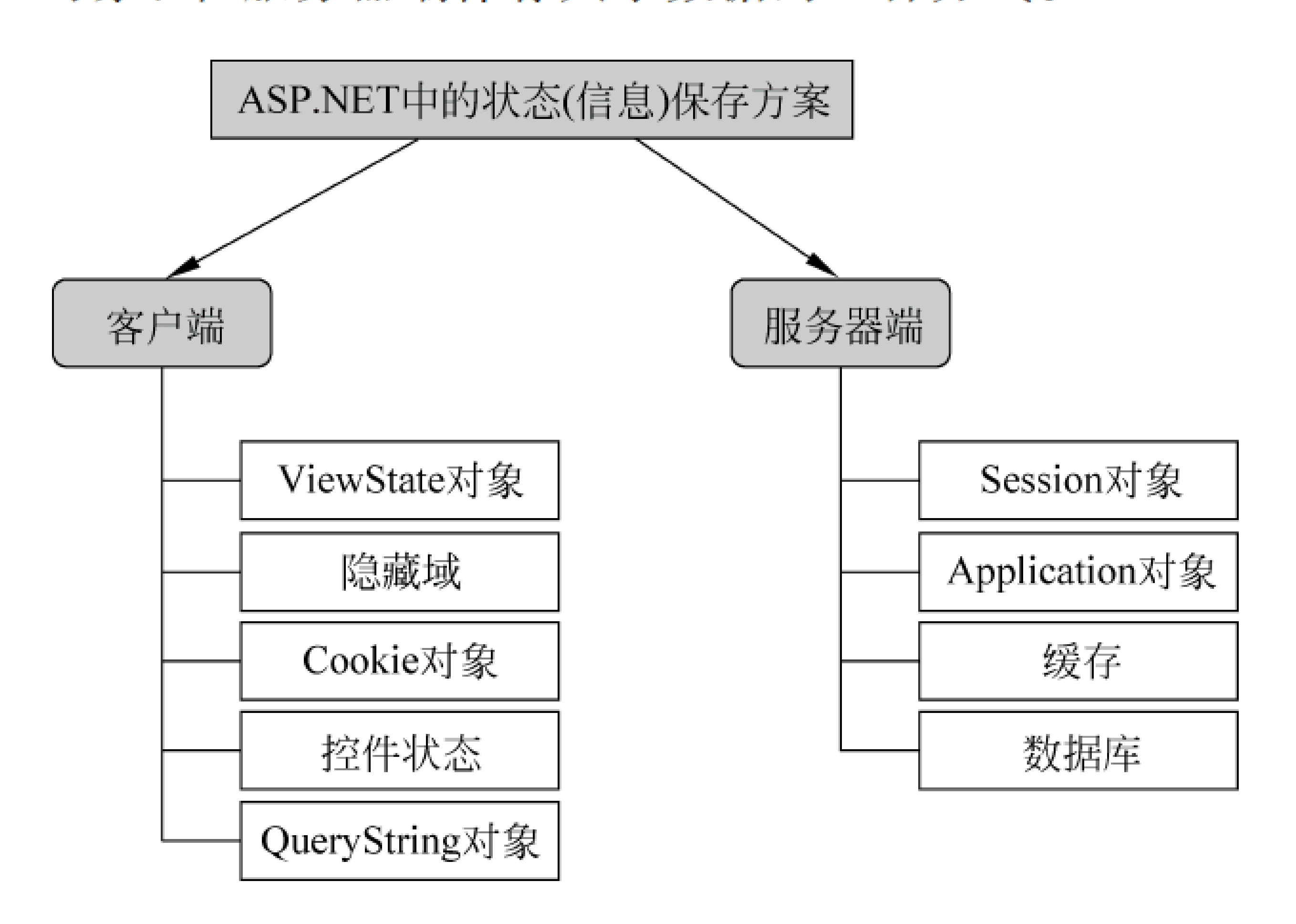


图 8.21　ASP .NET 中的状态(信息)保存方案

ViewState 对象不能存储所有的数据类型，仅支持 String、Integer、Boolean、Array、ArrayList 和 Hashtable。使用 ViewState 对象的前提是，网页上必须有一个服务器端表单标记(<form runat="server">)。服务器在接收到用户请求一个网页后，会自动在请求报文中查找，看是否包含_VIEWSTATE 的隐藏域，如果有，则将中间的值解码后添加到页面的 ViewState 属性中。服务器在输出时，也会自动地将 ViewState 中的值添加到表单里名叫_VIEWSTATE 的隐藏域中。

ViewState 对象适用于同一个网页在不关闭的情况下多次与服务器交互，跨页面提交的_VIEWSTATE 不会被目标网页加载到网页的 ViewState 对象中。

ViewState 对象的常用属性及其说明如表 8.19 所示。ViewState 对象的常用方法及其说明如表 8.20 所示。

表 8.19 ViewState 对象的常用属性及其说明

属性	说 明
Count	获取视图状态对象中的状态项个数
Item	获取或设置在视图状态对象中存储项的值
Keys	获取表示视图状态对象中项的键集合
Values	获取存储在视图状态对象中视图状态值的集合

表 8.20 ViewState 对象的常用方法及其说明

方法	说 明
Add	将新的状态项对象添加到 StateBag 对象。如果该项已经存于视图状态对象中，则此方法会更新该项的值
Clear	从当前视图状态对象中移除所有项
Remove	将指定的密钥/值对从视图状态对象中移除

例如：

```
ArrayList myarr = new ArrayList(4);        //定义名为 myarr 的 ArrayList 对象
myarr.Add("item 1");                       //向 myarr 中添加 4 个元素
myarr.Add("item 2");
myarr.Add("item 3");
myarr.Add("item 4");
ViewState.Add("mystate",myarr);            //将 myarr 存储到名为 mystate 的视图项中
ArrayList parr;                            //声明 parr
parr = (ArrayList)ViewState["mystate"];    //读取名为 mystate 的视图项到 parr 中
```

说明：视图状态信息是使用 Base64 编码存储的，并在呈现期间包括在网页中，这会增加网页大小。回发网页时，视图状态的内容作为网页回发信息的一部分发送。由于这会大大增加网络通信量和降低连接速度，因此建议不要在视图状态中存储大量信息。

8.9.2 ViewState 对象的应用

ViewState 对象的应用与 Application、Session 十分相似，只是将相关的键名和值存储在隐藏字段_VIEWSTATE 中。

【练一练】 在本章的网站 CH8 中添加一个 webform11 网页，其功能是说明 ViewState 对象的使用方法。

其设计步骤如下：

① 打开 CH8 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH8”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform11.aspx，其他保持默认项，单击“添加”按钮。

② 本网页的设计界面如图 8.22 所示，其中包含两个 HTML 文字、两个文本框(TextBox1 和 TextBox2、两个命令按钮(Button1 和 Button2)和一个标签 Label1。在该网页上设计如下事件过程：

```
protected void Button1_Click(object sender, EventArgs e)
{    if (TextBox1.Text != "" && TextBox2.Text != "")
```

```
        {   ViewState["xh"] = TextBox1.Text;
            ViewState["xm"] = TextBox2.Text;
            Label1.Text = "你的信息已保存到 ViewState";
        }
        else
            Label1.Text = "你的信息不完整,不能保存";
    }
    protected void Button2_Click(object sender, EventArgs e)
    {   if (ViewState["xh"]!= null)
        {   Label1.Text = "读取 ViewState 的信息如下:<br>";
            Label1.Text += "  学号为" + ViewState["xh"] +
                ",姓名为" + ViewState["xm"];
        }
        else
            Label1.Text = "你的信息没有保存,不能读出";
    }
```

③ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页,如果直接单击“读取 ViewState”按钮,其执行界面如图 8.23 所示。如果输入学号为 101,姓名为“王华”,单击“保存到 ViewState”按钮,其执行界面如图 8.24 所示。接着单击“读取 ViewState”按钮,其执行界面如图 8.25 所示。



图 8.22　webform11 网页的设计界面

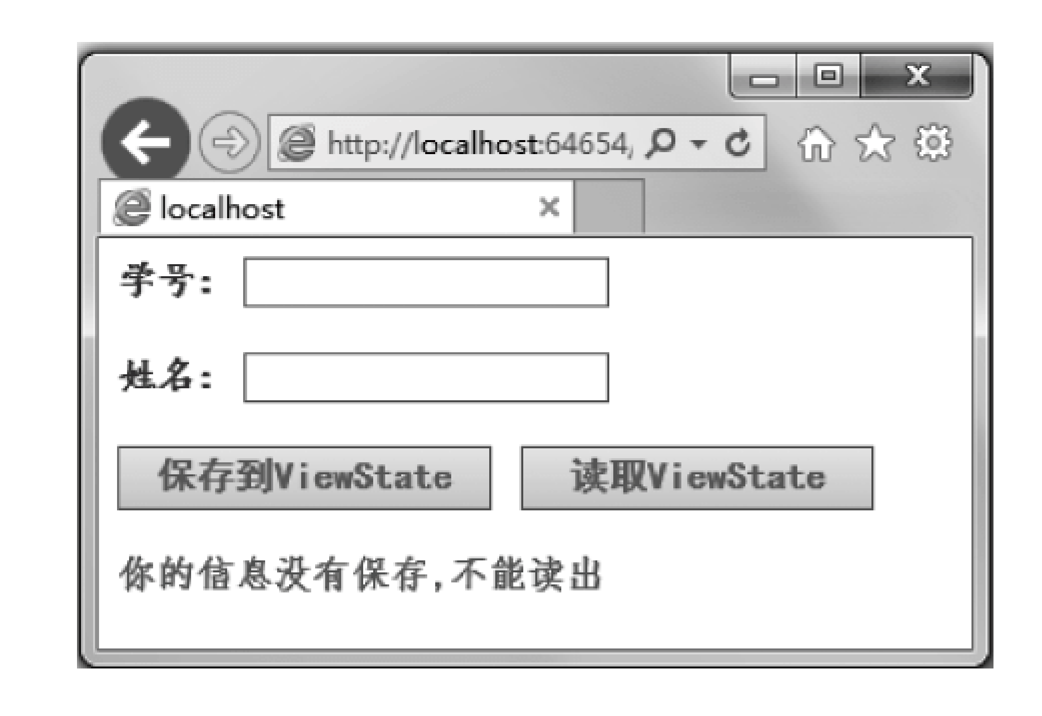


图 8.23　webform11 网页的执行界面一

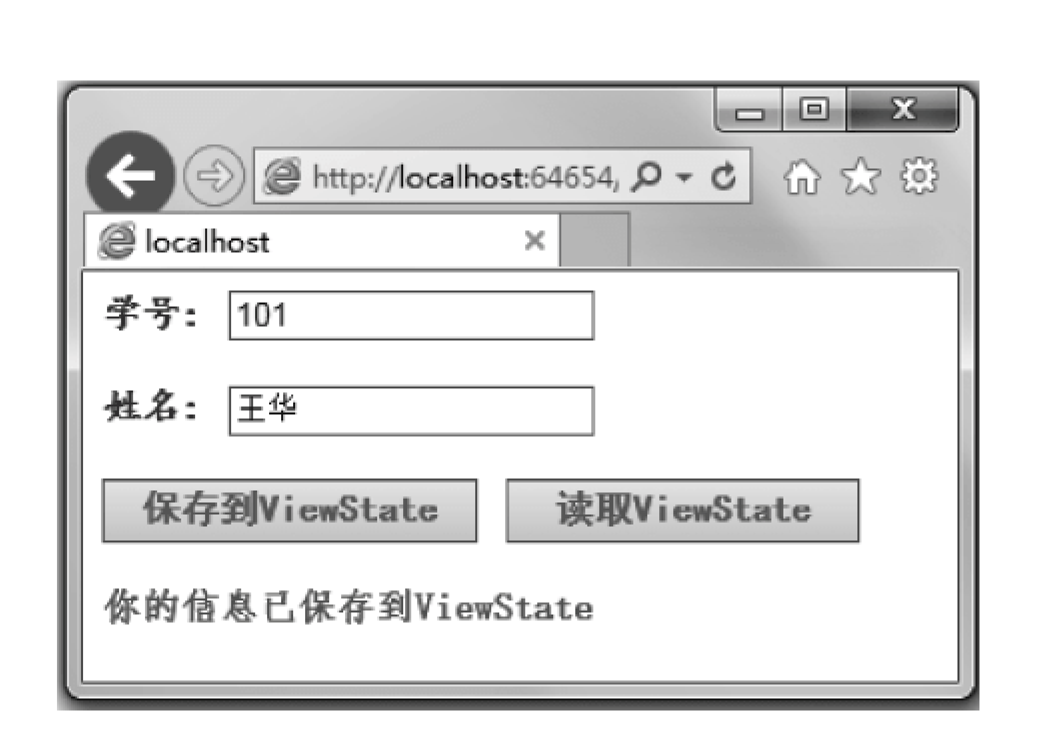


图 8.24　webform11 网页的执行界面二

图 8.25　webform11 网页的执行界面三

从执行结果可以看到 ViewState 对象的作用。

8.10 练　习　题

1. 单项选择题

(1) 在 ASP .NET 的内建对象中，必须要有 Lock 和 Unlock 方法以确保多个用户无法同时改变某一变量的对象是(　　)。

A. Cache　　B. Session　　C. Request　　D. Application

(2) 以下叙述中正确的是(　　)。

A. 当用户从一个网页转到另一个网页时，保存在 Session 中的信息会丢失

B. Session 对象的有效期默认为 40 分钟

C. Session 对象的有效期不能更改

D. Session 对象到期前可以用 Abandon 方法强行清除

(3) Request.Form 读取的数据是(　　)。

A. 以 Post 方式发送的数据　　B. 以 Get 方式发送的数据

C. 超链接后面的数据　　D. 以上都不对

(4) Response 对象的一个功能是实现从当前网页跳转到指定网页，其主要靠(　　)方法完成该功能。

A. Redirect()　　B. MapPath()　　C. End()　　D. Flush()

(5) Application 对象的默认有效期是(　　)。

A. 10 分钟　　B. 20 分钟

C. 30 分钟　　D. 从网站启动到终止

(6) 以下不属于 Request 对象集合成员的是(　　)。

A. Cookies　　B. Form　　C. QueryString　　D. Server

(7) Server 对象的 Excute 方法和 Transfer 方法的区别是(　　)。

A. 前者执行完调用网页，继续执行当前页面，后者不是

B. 前者执行完调用网页，不再继续执行当前页面，后者不是

C. 前者转移到调用的网页，执行新的页面，后者不是

D. 前者转移到调用的网页，不再执行当前的页面，后者不是

(8) 在执行 A 网页时，若要调用 B 网页，B 网页执行完后，继续执行 A 网页，则通过 Server 的(　　)方法来实现。

A. Transfer　　B. Redirect　　C. Execute　　D. href

(9) 在建立 Application 对象的时候会产生(　　)事件。

A. Application_OnStart　　B. Application_OnEnd

C. Application_OnCreate　　D. Application_OnNew

(10) 以下是使用 Application 对象时防止竞争的代码，空白处应为(　　)。

```
Application.lock();
Application["counter"] = (int)Application["counter"] + 1;
Application.________;
```

A. lock()　B. lock　C. unlock()　D. unlock

(11) 如果要在网页上添加一个计算器来统计访问人数的话，可以选用(　　)对象对计数变量 Count 的加法操作来实现。

A. Session　B. Application　C. Server　D. Page

(12) 执行完如下语句后，页面上显示的内容为(　　)。

```
Response.Write("A");
Response.End();
Response.Write("B");
```

A. A　B. AB　C. AC　D. ABC

(13) 请问下面程序段执行完毕，页面上显示内容是(　　)。

```
Response.Write("<a href = 'http://www.whu.edu.cn'>武汉大学</a>");
```

A. 武汉大学

B. <a href='http://www.whu.edu.cn'>

C. 武汉大学(超链接)

D. 该句有错，无法正常输出

(14) (　　)不是 ASP .NET 网页之间传递参数的方式。

A. 使用 QueryString　B. 使用 Session 变量

C. 使用 Server.Transfer　D. 使用 ViewState

(15) 如果 Session["a"]=1，Session["b"]=2，则 Session["a"] + Session["b"]的值是(　　)。

A. 12　B. 3　C. ab　D. 以上都不对

(16) 若设置 Session 的代码是 Session["greeting"]="hello wang!"，则取出该 Session 对象的语句是 string mystr=(　　)。

A. Session["greeting"]　B. Session["greeting"].ToString()

C. greeting　D. "greeting"

(17) 网页的有效期应该使用(　　)对象进行设置。

A. Session　B. Application　C. Response　D. Request

(18) 强行结束 Session，并清除会话中所有信息使用的方法是(　　)。

A. Clear　B. Abandon　C. Remove　D. 以上都不对

(19) Global.asax 文件中 Session_Start 事件在(　　)时激发。

A. 在每个请求开始时引发　B. 尝试对使用进行身份验证时引发

C. 启动会话时引发　D. 在应用程序启动时引发

(20) Session 与 Cookie 状态之间最大的区别在于(　　)。

A. 存储的位置不同　B. 类型不同

C. 生命周期不同　D. 容量不同

(21) 以下叙述中错误的是(　　)。

A. 当在网页上显示 HTML 标记时要通过 Server 对象的 HtmlEncode 方法编码再输出

B. Application 对象是一个公有变量，允许多个用户对它访问

C. Session 变量值可以在使用时随时读取

D. 使用 Server 对象的 MapPath 方法可以将指定的虚拟路径映射到服务器上相应的物理目录上

(22) 以下叙述中正确的是(　　)。

A. Server 对象提供了对客户机的访问技术

B. Cookies 功能是获取客户端浏览器的信息

C. Session.Timeout＝60 的语句的含义是 Session 会话有效期是 60s

D. 以上都不对

2. 问答题

(1) ASP .NET 中有哪些内置对象?

(2) 简述 Page.IsPostBack 属性和 Page.IsValid 属性的区别。

(3) 简述 Cookie 对象和 Session 对象的区别。

(4) 简述使用 Application 加锁与解锁。

(5) 列举常用的 ASP .NET 网页之间传递值的几种方式。

(6) Application、Session、ViewState 和 Cookie 对象都用于存储数据,哪些将数据存储在客户端? 哪些将数据存储在服务器端?

(7) 简述 ViewState 的优缺点。

(8) 一个网页中有一个命令按钮 Button1 和若干文本框,单击 Button1 时将其中所有文本框清空,编写该事件处理方法。

8.11　上机实验题

在 CH8 网站中添加一个 Exp 网页,在该网页中统计访问的人数。其执行界面如图 8.26 所示。

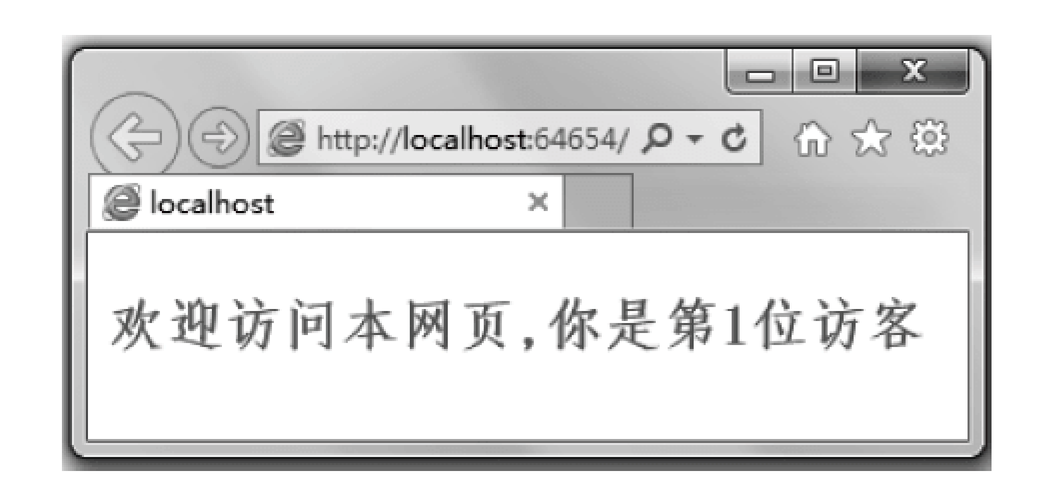
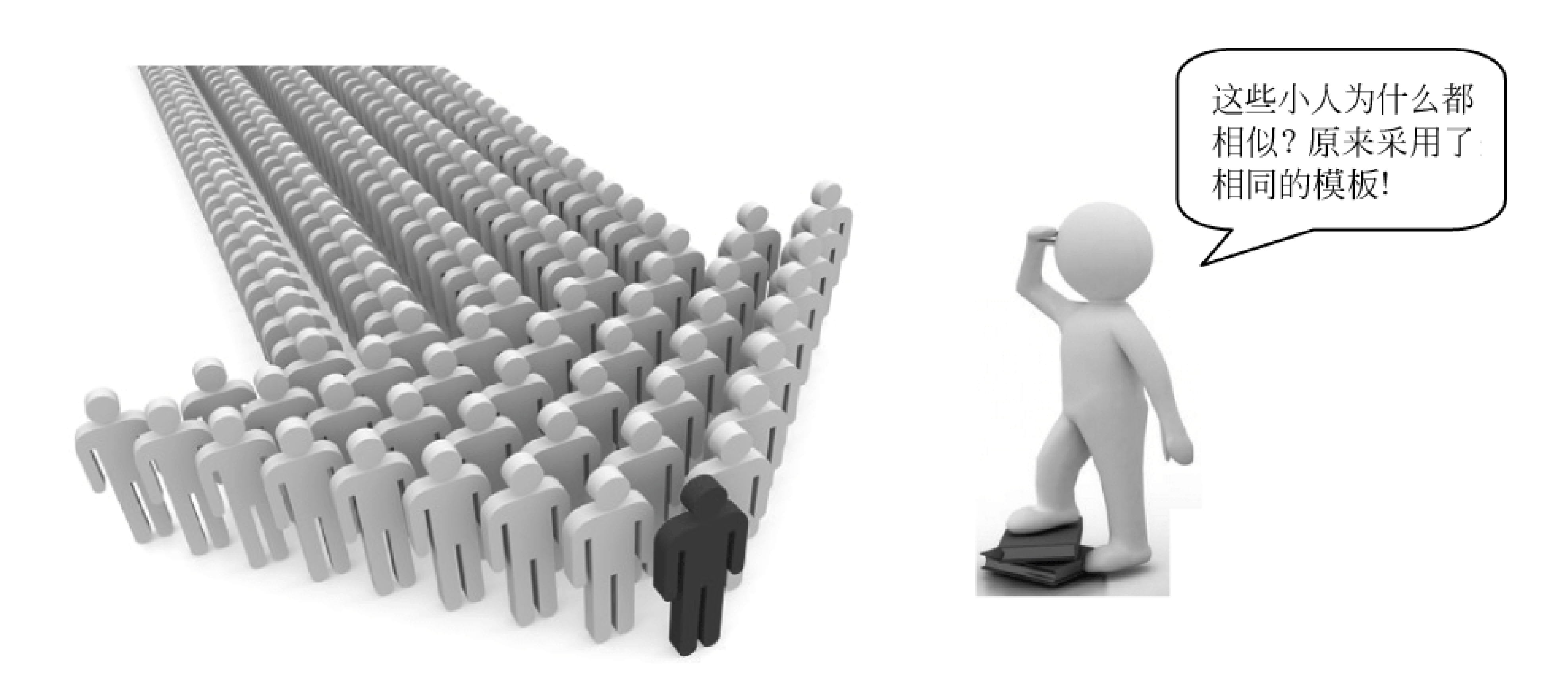


图 8.26　上机实验题网页的执行界面

第 9 章　主题、母版页和导航设计

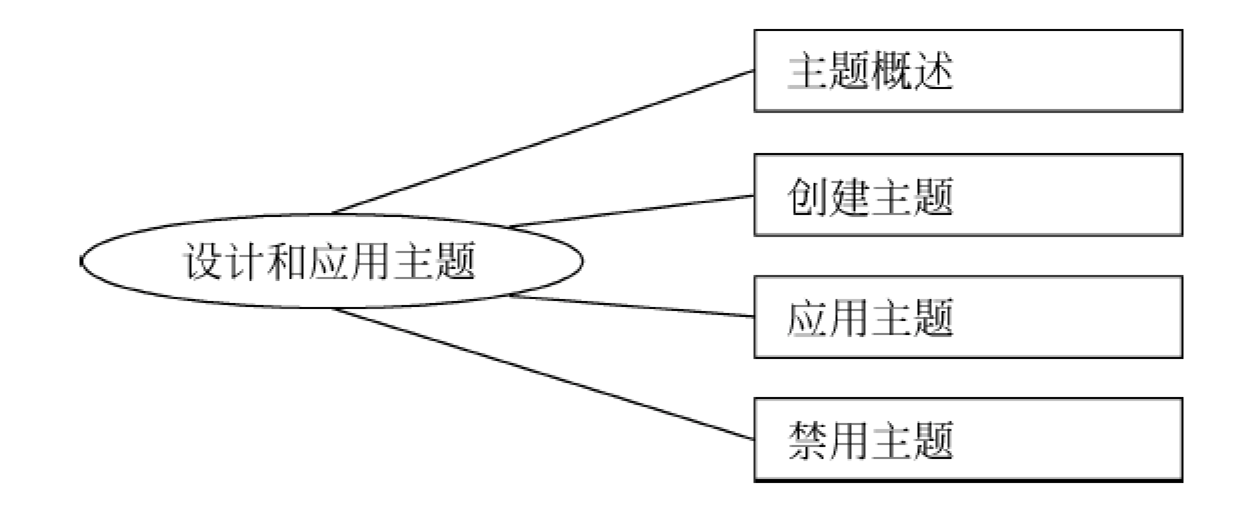


本章指南



9.1　设计和应用主题

知识梳理



9.1.1　主题概述

主题(theme)是指网页和控件外观属性设置的集合，其工作原理类似于 CSS，为网站提供

统一的风格。在CSS中，可以定义一个Link的样式，将这个样式存放在一个CSS样式文件中，然后在网站各网页中包含对这个文件的引用，结果网站中所有的Link样式便统一成这个文件中定义的那样了。

主题提供了一种简易方式，可以独立于应用程序的网页，为网站的控件和网页设置样式，因此便于Web应用程序对其进行维护。一个网站可以有多个主题，这样在设计网站时可以先不考虑样式，在以后要进行样式设计时，也无须更新网页或更改代码。另外，可以从外部获得自定义主题，如将另一个网站的主题复制到本网站中，因此主题可以方便地重用。

从实现上看，主题是存在于App_Themes目录中的子目录，每个子目录就是一个主题，其中可以包外观文件(.skin)和CSS文件(样式表文件)。在设计网页时不必在网页中显式引用主题，只需把它们放到App_Thems文件夹中，应用程序会自动加载相关的主题。

外观文件也称为皮肤文件，是主题的核心内容，用于定义网页中各种服务器控件(如Button、TextBox或Lable控件等)的外观属性。由一组控件的特定主题标记组成，其扩展名为skin文件。例如，如下代码设置Button控件的外观：

```
<asp:Button runat = "server" BackColor = "black" ForeColor = "Red" />
```

同一类型控件的外观分为默认外观和命名外观两种：

① 默认外观：不设置控件的SkinID属性，自动应用于同一类型的所有控件。在同一主题中只能有同一类型控件的一个默认外观，哪怕同一主题下有多个外观文件，但同一类型控件的默认外观也只能有一个。

② 命名外观：通过设置控件的SkinID属性，将命名外观应用于服务器控件，解决同一控件有多种属性设置的问题。例如，前面的代码指定命令按钮的默认外观，应用于网页中所有命令按钮。而以下代码属命名外观：

```
<asp:Button SkinID = "buttonskin1" runat = "server" BackColor = "gray" />
<asp:Button SkinID = "buttonskin2" runat = "server" BackColor = "white" />
```

这样为命令按钮设置了两种外观，在应用时需指定命令按钮控件的SkinID属性是buttonskin1或buttonskin2，这就是为什么几乎所有控件都有SkinID属性的原因。

9.1.2　创建主题

1. 创建主题的过程

【练一练】　以“D:\电子商务\CH9”目录创建文件系统空网站CH9，添加一个主题Theme1及其SkinFile.skin外观文件。

其设计步骤如下：

① 打开CH9网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH9(1)”对话框，在中间列表中选择“外观文件”，保持默认将文件名SkinFile.skin，如图9.1所示，单击“添加”按钮。

② 系统会自动创建一个位于网站根目录下的App_Themes文件夹，并在该文件夹中创建一个待命名的主题(默认名称为SkinFile)及外观文件SkinFile.skin。

③ 右击默认的主题名称SkinFile，在出现的快捷菜单中选择“重命名”命令，将主题名称改为Theme1。

④ 双击外观文件SkinFile.skin，打开它就可以编辑控件外观。例如，创建控件外观如下：

```
<asp:Label runat = "server" Font - Bold = "True" Font - Names = "仿宋"
    Font - Size = "Medium" ForeColor = "Fuchsia" />
<asp:Button runat = "server" Font - Bold = "True" Font - Names = "黑体"
    Font - Size = "Medium" ForeColor = "Red" />
<asp:TextBox runat = "server" Font - Names = "宋体" Font - Size = "Medium"
    ForeColor = "Black" BackColor = "White" />
```

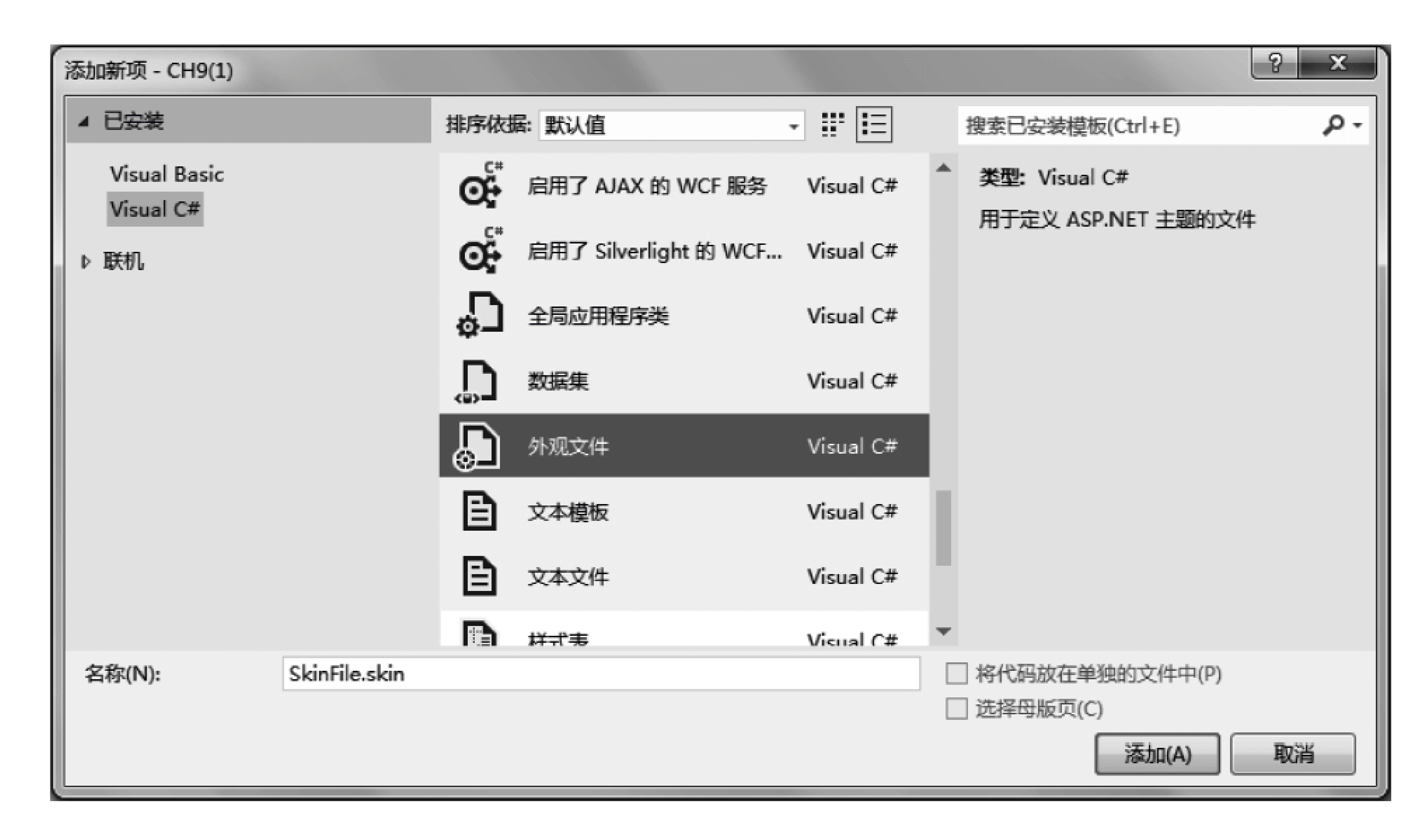


图 9.1　“添加新项”对话框

⑤ 单击 按钮保存。

如果要建立其他的主题，在“解决方案资源管理器”中右击 App_Themes，在弹出的快捷菜单中选择“添加|外观文件”命令，指定项名称，单击“确定”按钮即建立另一个主题。

一个主题中可以有多个外观文件。在一个主题中创建另一个外观文件的方法是，右击主题名 Theme1，在弹出的快捷菜单中选择“添加新项”命令，打开“添加新项”对话框。选择“外观文件”模板，指定外观文件名 SkinFile1.skin。单击“添加”按钮为 Theme1 主题添加了另一个外观文件 SkinFile1.skin，如图 9.2 所示。

注意：尽管一个主题中可以包含多个外观文件，但系统是按主题应用的，会把多个皮肤文件合并在一起，把这些文件视为一个文件。

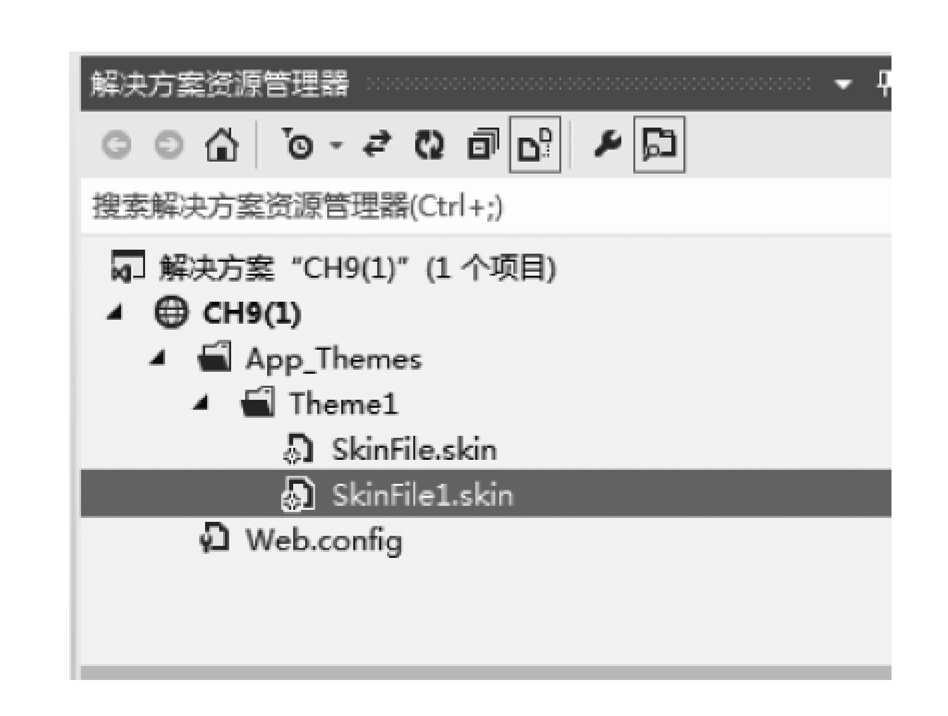


图 9.2　在 Theme1 主题下创建另一个外观文件

2. 外观设计方法

控件外观设计的常见方法是，先在网页中放置该类控件如 Lable1，然后可视化设计其外观，如对应的源视图代码如下：

```
<asp:Label ID = "Label1" runat = "server" Text = "Label" style = "color: #FF00FF;
    font - size: medium; font - weight: 700; font - family: 仿宋" />
```

再删除其中个性化的特定属性，如 ID、Text 属性等，得到如下外观：

```
<asp:Label runat="server" style="color: #FF00FF; font-size: medium;
    font-weight: 700; font-family: 仿宋" />
```

最后将上述代码放置在指定主题的外观文件中。

3. 外观文件的组织方式

通常外观文件的组织方式有以下 4 种。

- 无组织：将一个网站中所有控件属性设置放在一个外观文件中，对于初学者或小型网站可以采用这种方式。
- 根据控件类型组织：将同一类型控件的所有属性设置放在一个外观文件中，每种类型的控件对应一个外观文件，如所有文本框控件的外观放在一个外观文件中。
- 根据外观 SkinID 属性组织：将具有相同 SkinID 属性的外观放在一个外观文件中，每个 SkinID 属性值对应一个外观文件。
- 根据网页组织：将网站中每个网页的属性设置放在一个外观文件中，每个网页对应一个外观文件。这种方式很少使用。

9.1.3　应用主题

1. 指定网页的主题

常见的指定主题的方式有以下几种。

(1) 在网页的页指令中指定主题

将网页 Theme 属性设置为指定的主题。其操作是在"属性"窗口中指定 DOCUMENT 的 Theme 属性为指定的主题。这样网页的页指令自动变为：

```
<%@ Page Theme="Blue" … %>
```

也可以直接在页指令中添加上述粗体属性。

这种方式指定主题的作用在设计时不会显现，只有在网页运行时外观文件的作用来显现出来。

(2) 在代码中指定主题

可以在代码中为本网页指定主题，但需要放在 Page_PreInit 事件处理方法中才会生效，其一般格式如下：

```
protected void Page_PreInit()
{
    Page.Theme = "主题名";
}
```

(3) 在 web.config 文件中指定主题

与前两种方法相比，这是一种一劳永逸的方法，只需在 web.config 文件中＜pages＞节中定义 theme 属性，便可应用于整个网站。例如：

```
<configuration>
    <system.web>
        <pages theme = "主题名"></pages>
    </system.web>
</configuration>
```

（4）设置网页的 StyleSheetTheme 属性指定样式表主题

将网页 StyleSheetTheme 属性设置为指定的主题，这样指定的主题称为样式表主题。其操作是在“属性”窗口中指定 DOCUMENT 的 StyleSheetTheme 属性为指定的主题。这样网页的页指令自动变为：

```
<%@ Page Language = "C#" StyleSheetTheme = "Blue" … %>
```

也可以直接在页指令中添加上述粗体属性。

样式表主题和 Theme 属性指定的主题的区别是，后者只有在运行时外观才呈现，而前者在设计时外观便立即呈现。

上述方式都是为整个网页指定主题，网页中的控件会使用主题中该类型控件的默认外观；如果主题的控件外观带有 SkinID 属性，则还需要为特定的控件指定 SkinID 属性值。例如，在图 9.2 的 Theme1 主题 SkinFile1.skin 皮肤文件中输入如下外观：

```
<asp:Button SkinID = "buttonskin1" runat = "server" BackColor = "Black" Font - Bold = "True"
   ForeColor = "White" Font - Names = "隶书" Font - Size = "Medium" />
<asp:Button SkinID = "buttonskin2" runat = "server" BackColor = "White" Font - Bold = "True"
   ForeColor = "Red" Font - Names = "华文新魏" Font - Size = "Large" />
```

在一个网页上放置一个 Button1 命令按钮，设置其外观的 3 种情况如图 9.3 所示。主题 Theme1 包含两个皮肤文件，情况①使用的是 SkinFile.skin 中 Button 的默认外观，情况②使用的是 SkinFile1.skin 中命名外观 button1skin1，情况③使用的是 SkinFile1.skin 中命名外观 button1skin2。

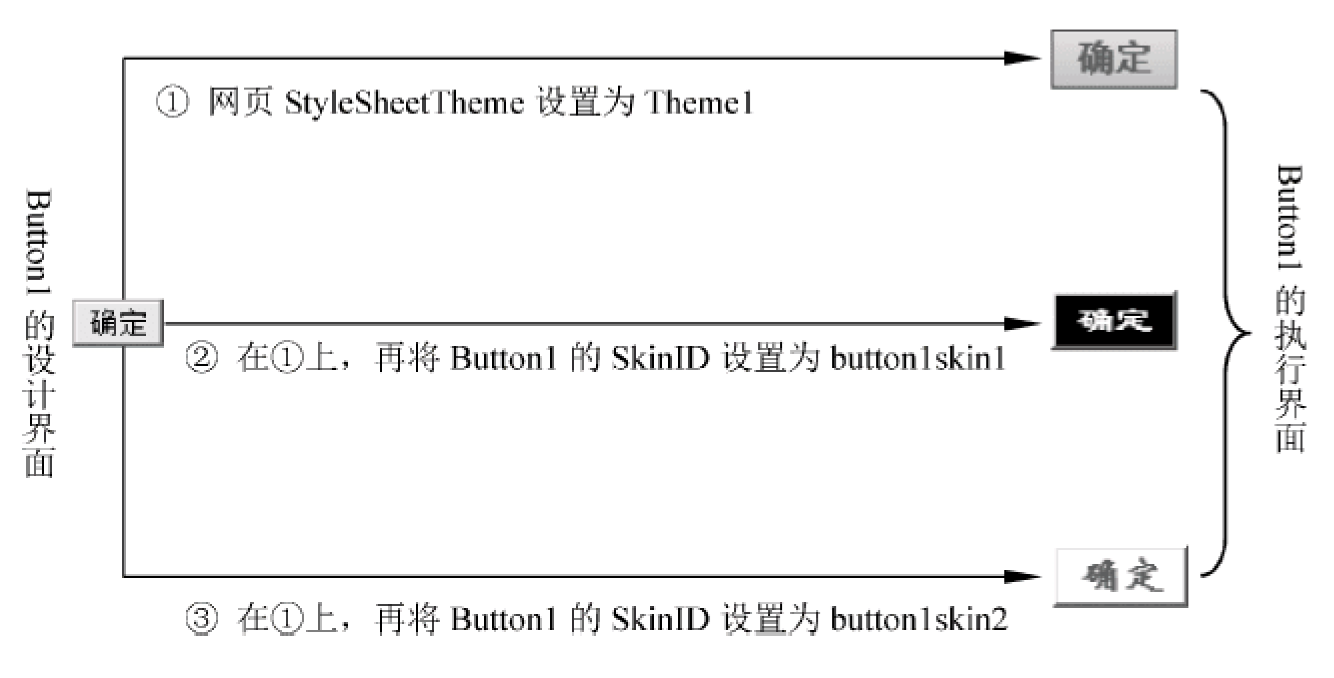


图 9.3　将外观应用于控件

2. 控件属性的应用顺序

如果对网页既设置 Theme 属性又设置 StyleSheetTheme 属性，则按以下顺序应用控件的属性：

① 首先应用 StyleSheetTheme 属性。

② 然后应用网页中的控件属性（重写 StyleSheetTheme）。

③ 最后应用 Theme 属性（重写控件属性和 StyleSheetTheme）。

【练一练】　在 CH9 网站中添加一个 webform1 网页，其功能是说明控件属性的应用顺序。

其设计步骤如下：

① 打开 CH9 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH9”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform1. aspx，其他保持默认项，单击“添加”按钮。

② 本网页的设计界面如图 9.4 所示，其中两个 HTML 文字标记、两个文本框（TextBox1 和 TextBox2）、一个命令按钮 Button1 和一个标签 Label1。将其 StyleSheetTheme 属性设为 Theme1，此时设计界面变为图 9.5 所示的样子，从而看到主题 Theme1 立即产生了作用。




图 9.4　webform1 网页的初始设计界面



图 9.5　设置 webform1 网页的初始设计界面

③ 在该网页上设计如下事件过程：

```
protected void Button1_Click(object sender, EventArgs e)
{   Label1.Text = "输入信息如下:<br>";
    Label1.Text += " 学号为" + TextBox1.Text + ",姓名为" + TextBox2.Text;
}
```

④ 在 CH9 网站中添加一个 Theme2 主题，其中包含一个外观文件 SkinFile. skin，输入外观代码如下：

```
<asp:Label runat="server" Font-Bold="True" Font-Names="仿宋"
    Font-Size="Medium" ForeColor="Fuchsia" />
<asp:Button runat="server" Font-Bold="True" Font-Names="黑体"
    Font-Size="Medium" ForeColor="Red" />
<asp:TextBox runat="server" Font-Names="宋体"
    Font-Size="Medium"
    ForeColor="Black" BackColor="ButtonFace"
    BorderStyle="Inset" BorderWidth="5px"/>
```

其中，包含有 Label、Button 和 TextBox 控件的默认外观。

⑤ 将本网页的 Theme 属性设为 Blue，此时设计界面没有变化。

⑥ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，输入学号和姓名，单击“确定”按钮，其结果如图 9.6 所示。从中看到 Theme2 主题中的外观。



图 9.6　webform1 网页的执行界面

9.1.4　禁用主题

1. 单个网页禁用主题

在网页的页指令中通过设置 EnableTheming 属性为 false 来使本网页禁用主题，其一般

格式如下：

```
<%@ Page EnableTheming = "主题名" … %>
```

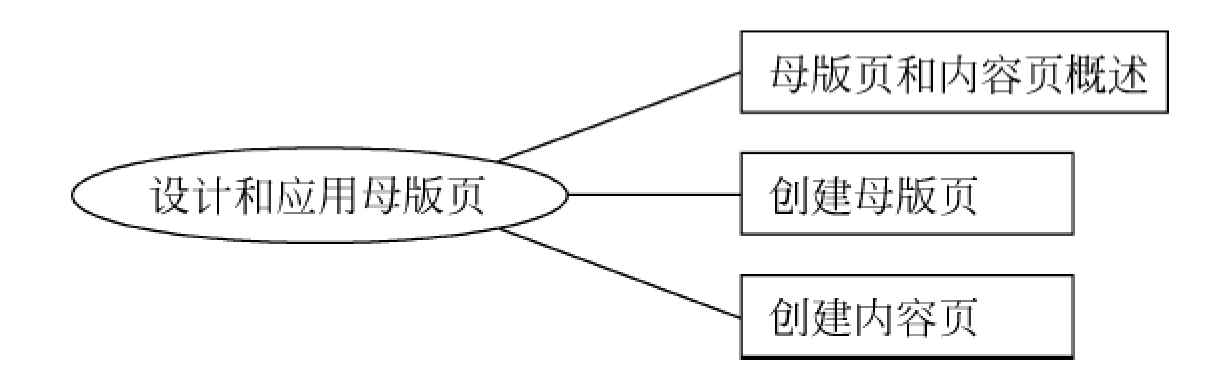
2. 单个网页中单个控件禁用主题

如果要使网页某个控件禁用主题，只需把这个控件的 EnableTheming 属性设置为 false 即可。例如，以下代码使得 Button1 控件禁用主题：

```
<asp:Button ID = "Button1" runat = "server" EnableThemeing = "false" />
```

9.2　设计和应用母版页

知识梳理



9.2.1　母版页和内容页概述

在设计网页时经常会遇到多个网页部分内容相同的情况，如果每个网页都设计一次显然是重复劳动且非常繁琐，为此 ASP .NET 提供了母版页来解决这个问题。母版页提供了统一管理和定义网页的功能，使多个网页具有相同的布局风格，给网页设计和修改带来很大方便。

1. 母版页

母版页是指其他网页可以将其作为模板来引用的特殊网页。母版页的扩展名为 master。在母版页中，界面被分为公用区和可编辑区，公用区的设计方法与一般网页的设计方式相同，可编辑区用 ContentPlaceHolder 控件预留出来，ContentPlaceHolder 控件起到占位符的作用，它在母版页中标识某个区域，该区域将预留给内容页。一个母版页中可以有一个可编辑区，也可以有多个可编辑区。

2. 内容页

引用母版页的.aspx 网页即为内容页。在内容页中，母版页的 ContentPlaceHolder 控件预留的可编辑区会被自动替换为 Content 控件，开发人员只需在 Content 控件区域中填充内容即可，在母版页中定义的其他标记将自动出现在使用了该母版页的.aspx 页面中。

3. 母版页和内容页的关系

在网页运行时，母版页和内容页的页面内容组合到一起，母版页中的占位符包含内容页中的内容，最后将完整的网页发送到客户浏览器。母版页和内容页的关系如图 9.7 所示。

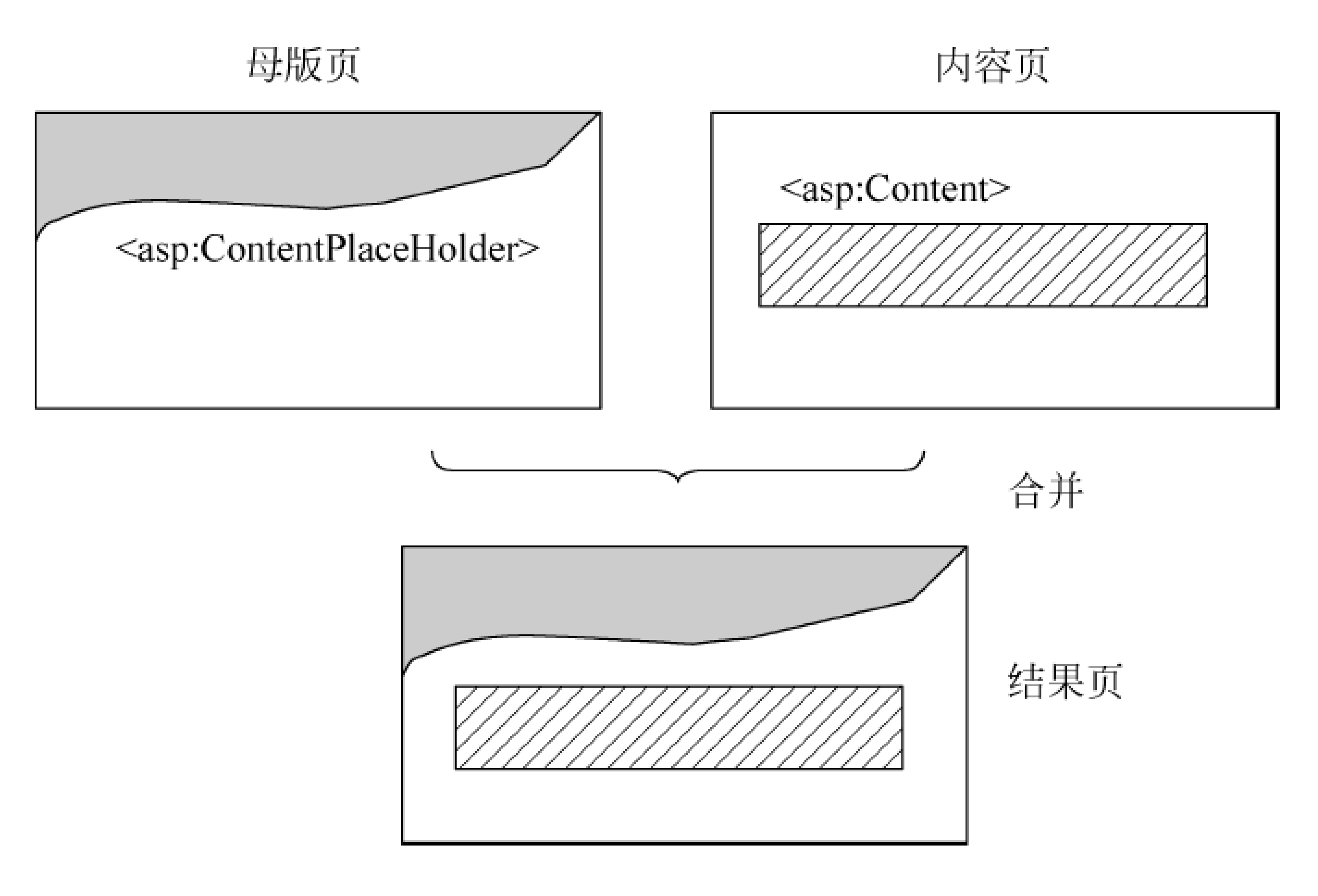


图 9.7　母版页和内容页的关系

9.2.2　创建母版页

母版页中包含的是网页的公共部分，即网页模板，因此在创建之前，必须确定哪些是页面的公共部分，需要从分析页面结构开始。

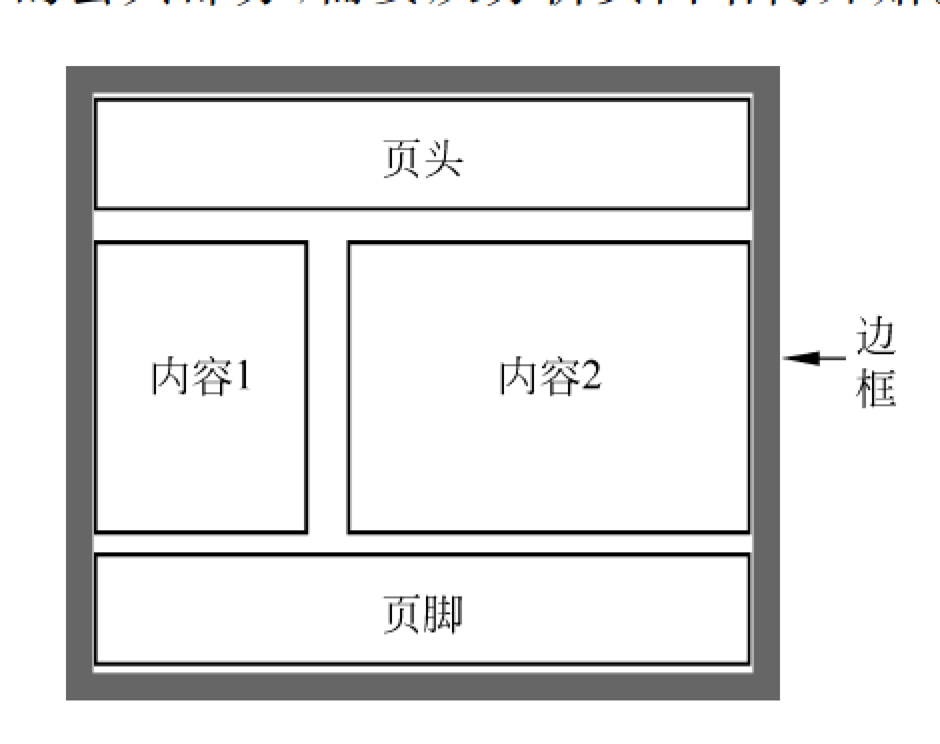


图 9.8　页面结构图

通常网页页面由边框、页头、内容和页脚组成，内容可能分为几个部分，如图 9.8 所示。一般地，网站中许多网页都包含相同的边框、页头和页脚，所以它们是网页的公共部分。而内容是网页的非公共部分，是一个网页独有的。

在 ASP .NET 中，母版页中非公共部分用 ContentPlaceHolder 控件表示。该控件取得内容占位符的作用，用于在母版页中定义相对内容区域。

创建母版页的过程与一般网页相似，区别仅仅是不能单独在浏览器中查看母版页，而必须通过内容页在浏览器中查看。

【练一练】 在 CH9 网站中添加一个母版页 MasterPage. master，用于后面设计登录的内容页。

其设计步骤如下：

① 打开 CH9 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH9”对话框，在中间列表中选择“母版页”，保持默认文件名 MasterPage. master，单击“添加”按钮，如图 9.9 所示。

② 出现母版页设计界面，设计网页模板。删除其中自动产生的 ContentPlaceHolder 控件，插入一个 3×3 的表格，在第 1 和第 3 行输入 HTML 文字，分别作为页头和页脚，在第 2 行第 2 列中放置一个 ContentPlaceHolder 控件 ContentPlaceHolder1，其他单元格设置相同的背景颜色，如图 9.10 所示。

进入源视图，看到该母版页的代码如下：

```
<%@ Master Language = "C#" AutoEventWireup = "true" CodeFile = "MasterPage.master.cs"
```

```
    Inherits = "MasterPage" %>
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head runat = "server">
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8"/>
    <title></title>
    <asp:ContentPlaceHolder id = "head" runat = "server">
    </asp:ContentPlaceHolder>
    <style type = "text/css">
        .auto-style1 {
            width: 500px; height: 300px;
            border-collapse:collapse; padding: 0px;
        }
        .auto-style2 {
            height: 75px; width:100%;
            background-color: #469B9D; text-align: center;
            font-family: 华文新魏; font-size: xx-large;
            color: #FF0000; font-weight: bold;
        }
        .auto-style3 {
            width: 500px; height:50px;
            font-family: 幼圆; font-weight: bold;
            font-size: large; color: #FF0000;
            text-align: center;
        }
    </style>
  </head>
  <body>
    <form id = "form1" runat = "server">
      <div>
        <table border = "0" class = "auto-style1">
            <tr>
               <td class = "auto-style2" colspan = "3" rowspan = "0">电子商务系统</td>
            </tr>
            <tr>
               <td rowspan = "2" style = "background-color: #469B9D"> </td>
               <td>
                 <asp:ContentPlaceHolder ID = "ContentPlaceHolder1" runat = "server">
                 </asp:ContentPlaceHolder>
               </td>
               <td rowspan = "2" style = "background-color: #469B9D"> </td>
            </tr>
            <tr>
               <td style = "background-color: #469B9D" class = "auto-style3">
                    版权所有 2013-2020</td>
            </tr>
        </table>
      </div>
    </form>
  </body>
</html>
```

从中看到，除了将 Page 页指令改为 Master 指令以及使用 ContentPlaceHolder 控件外，上述代码与一般的网页十分类似。

注意：母版页不支持主题设置，但可以在内容页上设置主题。

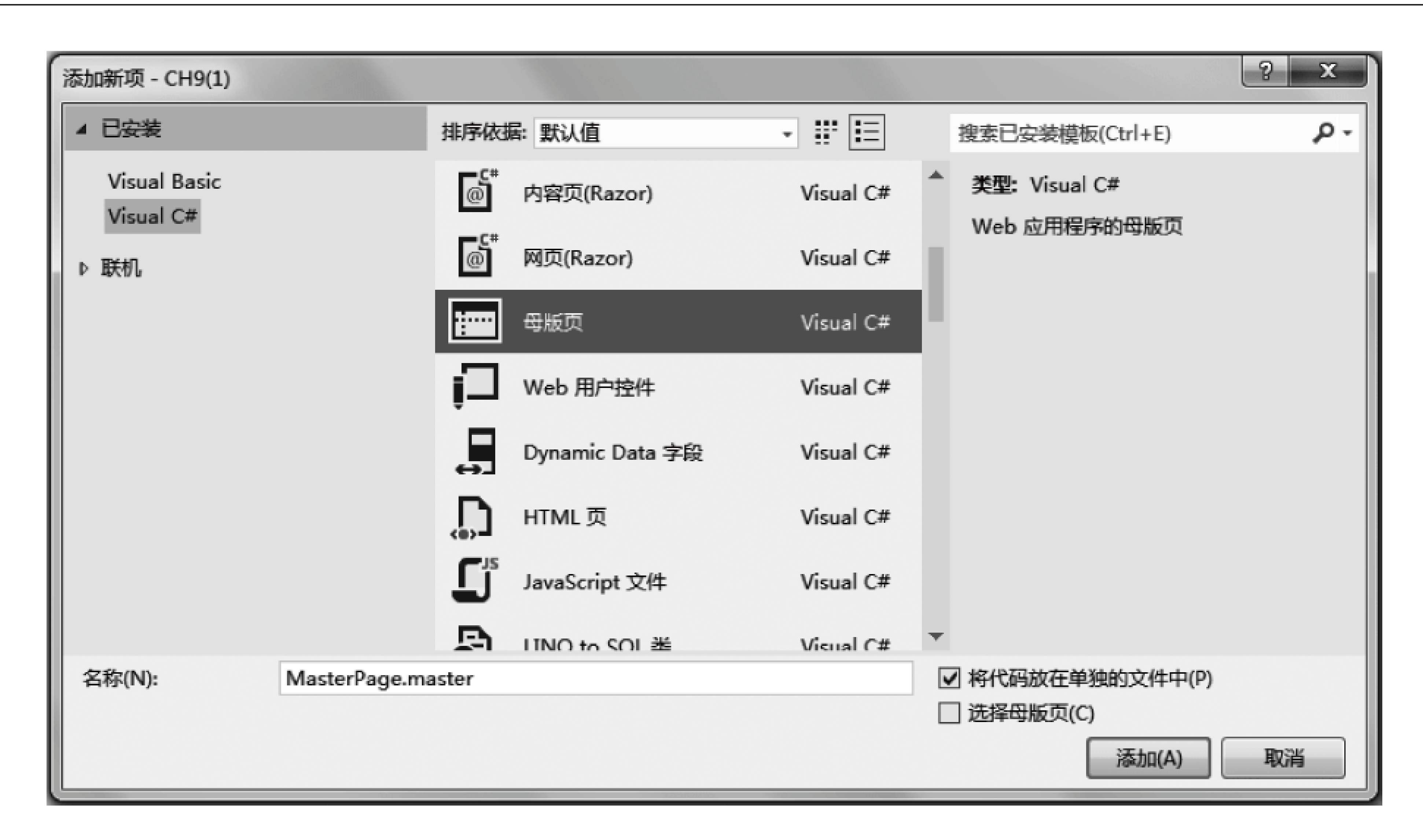


图 9.9 “添加新项”对话框



图 9.10 MasterPage.master 母版页设计界面

9.2.3 创建内容页

在创建一个完整的母版页之后，接下来必然根据母版页创建内容页，主要有两种方法：

- 在创建网页的“添加新项”对话框中勾选“选择母版页”复选框；
- 在母版页中右键，在弹出的快捷菜单中选择“添加内容页”命令。

在母版页中设计的 ContentPlaceHolder 控件，在内容页中对应 Content 控件。Content 控件使用其 ContentPlaceHolderID 属性与 ContentPlaceHolder 关联。将 ContentPlaceHolderID 属性设置为母版页中相关的 ContentPlaceHolder 控件的 ID 属性的值。例如：

```
<asp:Content ID = "Content2" ContentPlaceHolderID = "ContentPlaceHolder1"
    Runat = "Server"> </asp:Content>
```

上述语句表示内容页中 Content2 控件与母版页中 ContentPlaceHolder1 控件关联。

另外，内容页必须绑定到母版页，其方式是在内容页的页指令中设置 MasterPageFile 属

性为指定的母版页。例如，@Page 页面指令包含如下属性定义表示本内容页的母版页是 MasterPage.master 文件：

```
MasterPageFile = "~/MasterPage.master"
```

【练一练】 在 CH9 网站中添加一个内容页 webform2，其母版页为 MasterPage.master。其设计步骤如下：

① 打开 CH9 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH9(1)”对话框，在中间列表中选择“Web 窗体”，修改文件名为 webform2.aspx，勾选“选择母版页”，单击“添加”按钮，如图 9.11 所示。



图 9.11　“添加新项”对话框

② 出现如图 9.12 所示的“选择母版页”对话框，选中 MasterPage.master，单击“确定”按钮。

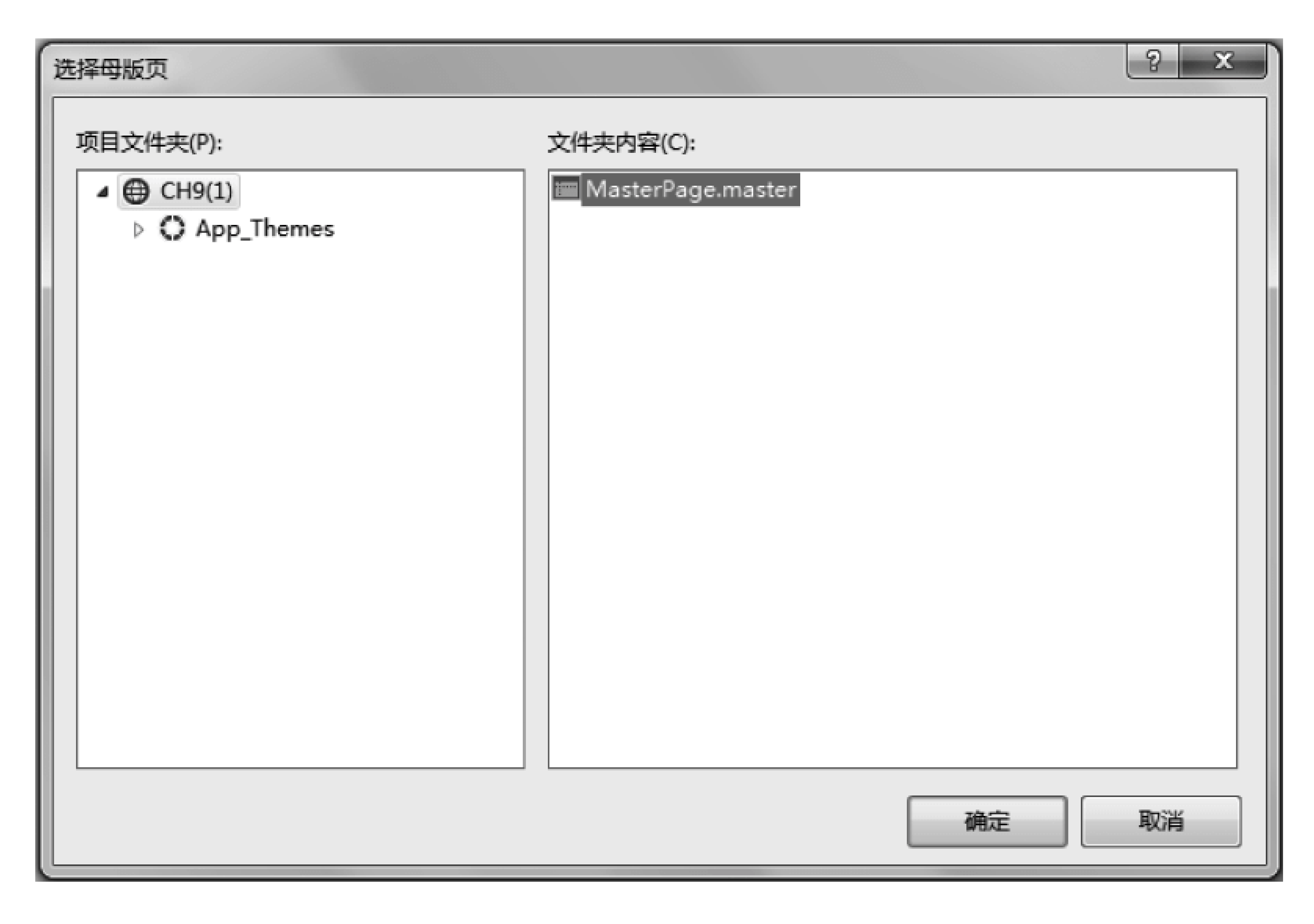


图 9.12　“选择母版页”对话框

③ 出现 webform2 内容页如图 9.13 所示的设计界面，除了对应母版页中 ContentPlaceHolder1 控件的区域外，其他部分是只读不可编辑的。



图 9.13　webform2 内容页的初始设计界面

在本网页的可编辑区域中插入一个 5 行 2 列的表格，再添加相关控件，最终设计的界面如图 9.14 所示。



图 9.14　webform2 内容页的最终设计界面

本网页的源视图代码如下：

```
<%@ Page Title = "" Language = "C#" MasterPageFile = "~/MasterPage.master"
    AutoEventWireup = "true"
    CodeFile = "webform2.aspx.cs" Inherits = "webform2" %>
<asp:Content ID = "Content1" ContentPlaceHolderID = "head" Runat = "Server">
  <style type = "text/css">
    .auto-style4 {
        width: 100%; }
    .auto-style5 {
            text-align: center; font-family: 隶书;
            font-weight: bold; font-size: x-large;
            color: #800080;
    }
    .auto-style6 {
            font-family: 楷体;font-size: medium;
            color: #0000FF; font-weight: bold;
            text-align: right;width: 154px;
```

```
        }
    </style>
</asp:Content>
<asp:Content ID = "Content2" ContentPlaceHolderID = "ContentPlaceHolder1"
        Runat = "Server">
        <table class = "auto-style4">
            <tr>
                <td class = "auto-style5" colspan = "2">用户登录</td>
            </tr>
            <tr>
                <td class = "auto-style6">用户名</td>
                <td><asp:TextBox ID = "TextBox1" runat = "server"></asp:TextBox></td>
            </tr>
            <tr>
                <td class = "auto-style6">密   码</td>
                <td><asp:TextBox ID = "TextBox2" runat = "server"
                        TextMode = "Password"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td colspan = "2" style = "text-align: center">
                    <asp:Button ID = "Button1" runat = "server" Text = "登录"
                      style = "color: #FF0000; font-size: medium; font-weight: 700;
                         font-family: 黑体;" OnClick = "Button1_Click" />
                </td>
            </tr>
            <tr>
                <td colspan = "2" style = "text-align: center">
                    <asp:Label ID = "Label1" runat = "server" style = "color: #FF00FF;
                       font-size: medium; font-weight: 700; font-family: 仿宋"></asp:Label>
                </td>
            </tr>
        </table>
</asp:Content>
```

从中看到，内容页的源视图代码与一般网页的源视图代码不同，不包含<html>、<body>等 Web 元素，这些元素包含在母版页中。内容页的源视图代码主要由两部分组成：代码头声明和 Content 控件区。代码头声明中，页指令重要的属性是 MasterPageFile（设置所绑定的母版页）和 Title（设置网页标题）。Content 控件区中包含一个或多个 Content 控件，用来包含网页中的非公共内容。Content 控件通过 ContentPlaceHolderID 属性与母版页中的 ContentPlaceHolder 控件关联。

④ 在本网页上设计如下事件处理方法：

```
protected void Button1_Click(object sender, EventArgs e)
{    if (TextBox1.Text == "User1234" && TextBox2.Text == "1234")
         Label1.Text  =  "提示：你已成功登录!";
     else
         Label1.Text  =  "提示：你的登录信息不正确，登录失败!";
}
```

这里假设用户名和密码是固定的，正确的用户名/密码为 User1234/1234，其他都是错误的。

⑤ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，如果用户输入错误的用户名/密码，其执行界面如图 9.15 所示。如果用户输入正确的用户名/密码，其执行界面如图 9.16 所示。

这里仅仅介绍了母版页和内容页设计的基本方法，更多有关从内容页中访问母版页的内容等复杂知识可以参阅相关文献。



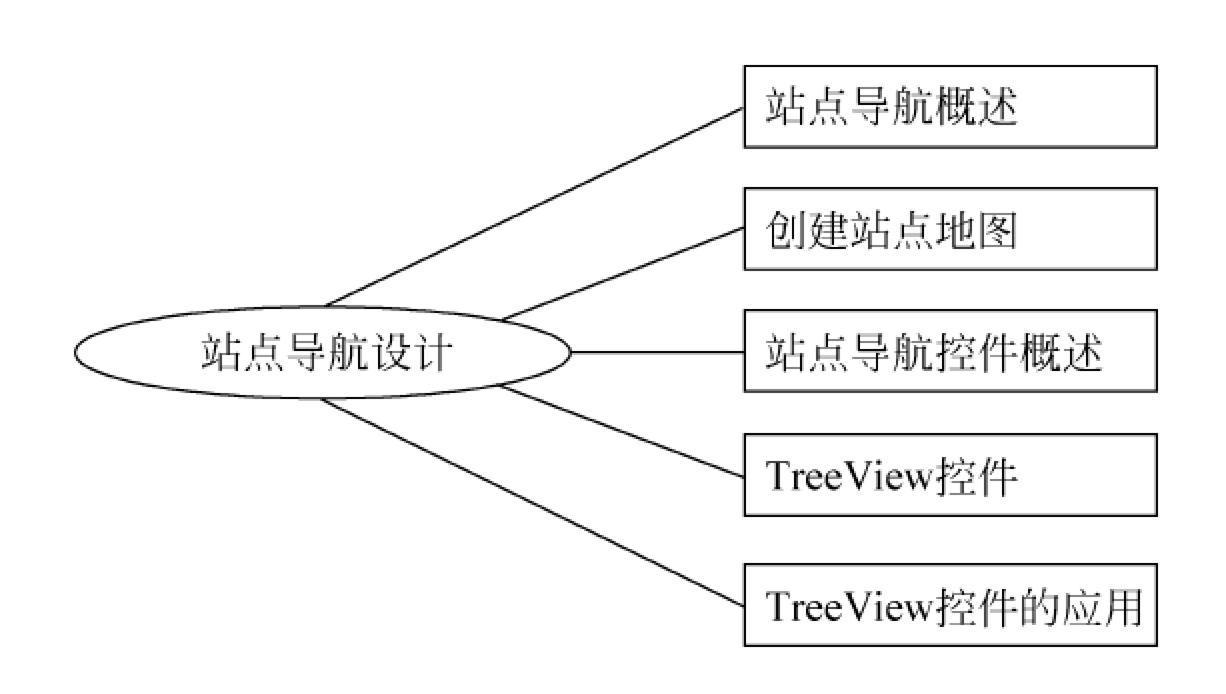
图 9.15　不成功登录时的界面



图 9.16　成功登录时的界面

9.3　站点导航设计

知识梳理



9.3.1　站点导航概述

使用 ASP .NET 站点导航功能可以为用户导航站点提供一致的方法。随着站点内容的增加以及用户在站点内来回移动网页，管理所有的链接会变得比较困难。ASP .NET 站点导航使用户能够将指向所有网页的链接存储在一个中央位置，并在列表中呈现这些链接，或用一个特定 Web 服务器控件在每个网页上呈现导航菜单。

设计站点导航功能主要使用 ASP .NET 提供下列功能：

- 站点地图：可以使用站点地图描述站点的逻辑结构，接着通过在添加或移除页面时修改站点地图(而不是修改所有网页的超链接)来管理页导航。
- ASP .NET 导航控件：可以使用 ASP .NET 控件在网页上显示导航菜单，导航菜单以站点地图为基础。

9.3.2　创建站点地图

若要使用站点导航，先创建一个站点地图或站点的表示形式。站点地图采用 XML 文件描述站点的层次结构。在创建站点地图后，可以使用站点导航控件在 ASP .NET 网页上显示其导航结构。

站点地图是一种以 sitemap 为扩展名的标准 XML 文件，主要为站点导航控件提供站点层次结构信息，默认名为 Web. sitemap。

【练一练】 在 CH9 网站中添加一个 Web. sitemap 站点地图，它反映一个大学的网站布局，如图 9.17 所示。

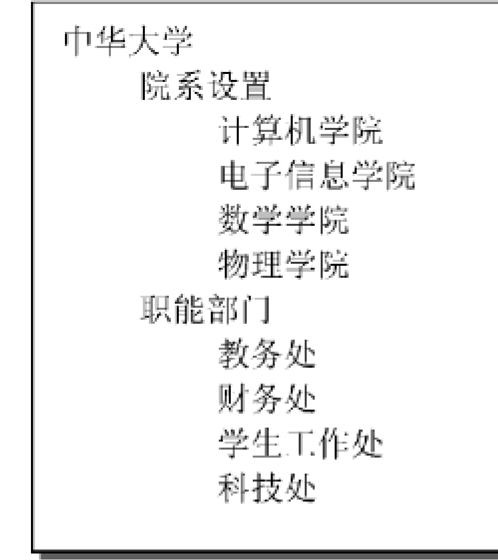


图 9.17　一个大学的网站布局

其设计步骤如下：

① 打开 CH9 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH9(1)”对话框，在中间列表中选择“站点地图”，保持默认名称为 Web. sitemap(只有名称为 Web. sitemap 的站点地图才会被自动加载，并且必须出现在网站的根目录中)，单击“添加”按钮，如图 9.18 所示。

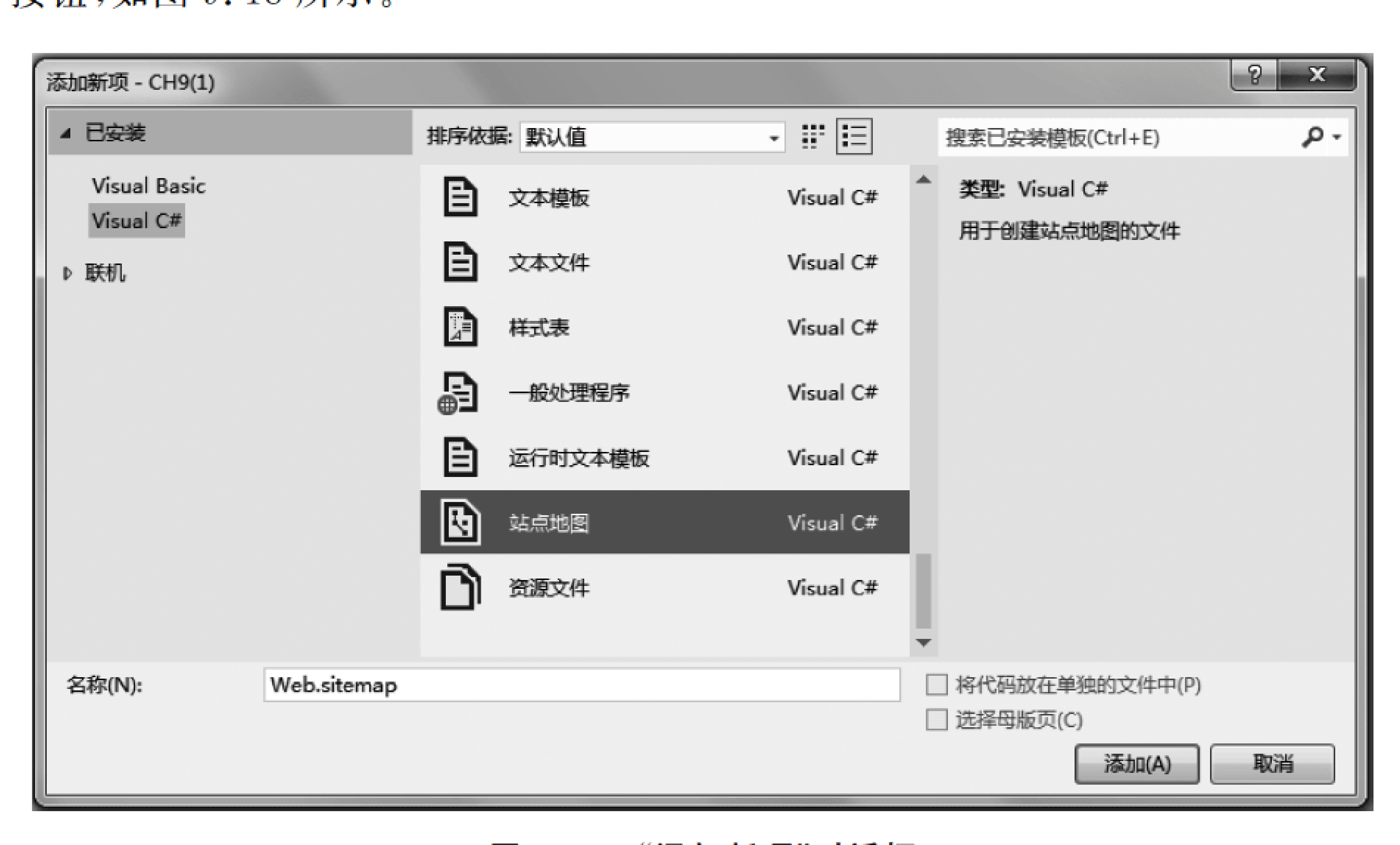


图 9.18　“添加新项”对话框

② 出现站点地图的编辑窗口，编辑站点地图 Web.sitemap 文件如下：

```
<?xml version = "1.0" encoding = "utf - 8" ?>
< siteMap xmlns = "http://schemas.microsoft.com/AspNet/SiteMap - File - 1.0" >
  < siteMapNode url = "disp.aspx?mc = 中华大学" title = "中华大学" description = "">
    < siteMapNode url = "disp.aspx?mc = 院系设置" title = "院系设置" description = "">
      < siteMapNode url = "disp.aspx?mc = 计算机学院" title = "计算机学院" description = "" />
      < siteMapNode url = "disp.aspx?mc = 电子信息学院" title = "电子信息学院" description = "" />
      < siteMapNode url = "disp.aspx?mc = 数学学院" title = "数学学院" description = "" />
      < siteMapNode url = "disp.aspx?mc = 物理学院" title = "物理学院" description = "" />
    </siteMapNode >
    < siteMapNode url = "disp.aspx?mc = 职能部门" title = "职能部门" description = "">
      < siteMapNode url = "disp.aspx?mc = 教务处" title = "教务处" description = "" />
      < siteMapNode url = "disp.aspx?mc = 财务处" title = "财务处" description = "" />
      < siteMapNode url = "disp.aspx?mc = 学生工作处" title = "学生工作处" description = "" />
      < siteMapNode url = "disp.aspx?mc = 科技处" title = "科技处" description = "" />
    </siteMapNode >
  </siteMapNode >
</siteMap >
```

③ 单击💾保存该站点地图文件。

站点地图是一个标准 XML 文件。其中，第一个标记用于标识版本和编码方式，siteMap 是站点地图根节点标记，包含若干个 siteMapNode 子节点，一个 siteMapNode 子节点下又可以包含若干个 siteMapNode 子节点，构成一种层次结构。

siteMapNode 节点的常用属性如表 9.1 所示。

表 9.1　siteMapNode 节点的常用属性

属　性	说　明
url	设置用于节点导航的 URL 地址。在整个站点地图文件中，该属性必须唯一
title	设置节点名称
description	设计节点说明文字
key	定义当前节点的关键字
roles	定义允许查找该站点地图文件的角色集合，多个角色可用分号(;)或逗号(,)分隔
Provider	定义处理其他站点地图文件的站点导航提供程序名称，默认为 XmlSiteMapProvider

9.3.3　站点导航控件概述

创建一个反映站点结构的站点地图只完成了 ASP .NET 站点导航系统的一部分。导航系统的另一部分是在 ASP .NET 网页中显示导航结构，这样用户就可以在站点内轻松地移动。通过使用下列 ASP .NET 站点导航控件，可以轻松地在页面中建立导航信息：

- TreeView：此控件显示一个树状结构或菜单，让用户可以遍历访问站点中的不同页面，单击包含子节点的节点可将其展开或折叠。
- Menu：此控件显示一个可展开的菜单，让用户可以遍历访问站点中的不同页面。将光标悬停在菜单上时，将展开包含子节点的节点。
- SiteMapPath：此控件显示导航路径(也称为面包屑或眉毛链接)向用户显示当前页面

的位置，并以链接的形式显示返回主页的路径。此控件提供了许多可供自定义链接的外观的选项。

所有站点导航控件均位于工具箱的"导航"选项卡中，可以像其他服务器控件一样使用。这里主要介绍 TreeView 控件。

9.3.4　TreeView 控件

TreeView 控件又称为树形导航控件。它的显示类似于一棵横向的树，可以展开或折叠树的节点来分类查看、管理信息，非常直观。

TreeView 控件由节点组成。树中的每个项都称为一个节点，它由一个 TreeNode 对象表示。节点类型的定义如下：

- 包含其他节点的节点称为父节点(Parent Node)。
- 被其他节点包含的节点称为子节点(Child Node)。
- 没有子节点的节点称为叶节点(Leaf Node)。
- 不被其他任何节点包含同时是所有其他节点的上级的节点是根节点(Root Node)。

一个节点可以同时是父节点和子节点，但是不能同时为根节点、父节点和叶节点。节点为根节点、父节点还是叶节点决定着节点的几种可视化属性和行为属性。

尽管通常的树结构只具有一个根节点，但是 TreeView 控件允许向树结构中添加多个根节点。如果要在不显示单个根节点的情况下显示项列表，这种控件就非常有用。

1. TreeNode 类

TreeView 控件中一个节点就是一个 TreeNode 类对象。TreeNode 类的常用属性如表 9.2 所示，常用方法如表 9.3 所示。

表 9.2　TreeNode 类的常用属性及其说明

属　性	说　明
Checked	获取或设置一个值，该值指示节点的复选框是否被选中
ChildNodes	获取 TreeNodeCollection 集合，该集合包含当前节点的第一级子节点
Depth	获取节点的深度
Expanded	获取或设置一个值，该值指示是否展开节点
ImageToolTip	获取或设置在节点旁边显示的图像的工具提示文本
ImageUrl	获取或设置节点旁显示的图像的 URL
NavigateUrl	获取或设置单击节点时导航到的 URL
Parent	获取当前节点的父节点
Selected	获取或设置一个值，该值指示是否选择节点
ShowCheckBox	获取或设置一个值，该值指示是否在节点旁显示一个复选框
Target	获取或设置用来显示与节点关联的网页内容的目标窗口或框架
Text	获取或设置为 TreeView 控件中的节点显示的文本
ToolTip	获取或设置节点的工具提示文本
Value	获取或设置用于存储有关节点的任何其他数据(如用于处理回发事件的数据)的非显示值
ValuePath	获取从根节点到当前节点的路径

表 9.3　TreeNode 类的常用方法及其说明

方　　法	说　　明
Collapse	折叠当前树节点
CollapseAll	折叠当前节点及其所有子节点
Expand	展开当前树节点
ExpandAll	展开当前节点及其所有子节点
Select	选择 TreeView 控件中的当前节点
ToggleExpandState	切换节点的展开和折叠状态

每个 TreeView 对象都具有一个 Text 属性和一个 Value 属性。Text 属性的值显示在 TreeView 控件中，而 Value 属性用于存储有关节点的任何其他数据，如传递到与该节点相关联的回发事件的数据。在 TreeView 控件，节点(即 TreeView 对象)可以处于以下两种状态之一：选定状态和导航状态。在默认情况下，会有一个节点处于选定状态(该节点的 Selected 属性为 true)。若要使一个节点处于导航状态，需将该节点的 NavigateUrl 属性值设置为空字符串以外的值。若要使一个节点处于选定状态，需将该节点的 NavigateUrl 属性值设置为空字符串。

TreeNode 类提供了以下构造函数：

```
public TreeNode()
public TreeNode (string text)
public TreeNode (string text,string value)
public TreeNode (string text,string value,string imageUrl)
public TreeNode (string text,string value,string imageUrl,string navigateUrl,string target)
```

其中，参数 text 指定 TreeView 控件中的节点显示的文本。value 指定与节点关联的补充数据，如用于处理回发事件的数据。imageUrl 指定节点旁显示的图像的 URL。navigateUrl 指定单击节点时链接到的 URL。target 指定单击节点时用来显示链接到的网页内容的目标窗口或框架。

2. TreeView 控件的属性、方法和事件

TreeView 控件的常用属性及其说明如表 9.4 所示。下面介绍几个主要的属性。

表 9.4　TreeView 控件的常用属性及其说明

属　　性	说　　明
CheckedNodes	获取 TreeNode 对象的集合，这些对象表示在 TreeView 控件中显示的选中了复选框的节点
DataSourceID	设置数据源对象
ExpandDepth	获取或设置第一次显示 TreeView 控件时所展开的层次数
NodeIndent	获取或设置 TreeView 控件的子节点的缩进量(以像素为单位)
Nodes	获取或设置 TreeNode 对象的集合，表示 TreeView 控件中节点
NodeStyle	获取对 TreeNodeStyle 对象的引用，该对象用于设置 TreeView 控件中节点的默认外观
NodeWrap	获取或设置一个值，指示空间不足时节点中的文本是否换行
RootNodeStyle	获取对 TreeNodeStyle 对象的引用，该对象用于设置 TreeView 控件中根节点的外观
SelectedNode	获取表示 TreeView 控件中选定节点的 TreeNode 对象

续表

属　　性	说　　明
SelectedValue	获取选定节点的值
ShowCheckBoxes	获取或设置一个值，指示哪些节点类型将在 TreeView 控件中显示复选框
ShowExpandCollapse	获取或设置一个值，指示是否显示展开节点指示符
ShowLines	获取或设置一个值，指示是否显示连接子节点和父节点的线条
CollapseImageUrl	可折叠节点的指示符所显示图像的 URL，此图像通常为一个减号(－)
ExpandImageUrl	可展开节点的指示符所显示图像的 URL，此图像通常为一个加号(＋)
LineImagesFolder	包含用于连接父节点和子节点的线条图像的文件夹的 URL。ShowLines 属性还必须设置为 true，该属性才能有效
NoExpandImageUrl	不可展开节点的指示符所显示图像的 URL

(1) DataSourceID 属性

该属性指定 TreeView 控件的数据源控件的 ID 属性。例如，可以指定与 XML 文件绑定的 XmlDataSource 控件或与站点地图绑定的 SiteDataSource 控件的 ID。

(2) ExpandDepth 属性

该属性获取或设置第一次显示 TreeView 控件时所展开的层次数。例如，若该属性设为 2，则将展开根节点及根节点下方紧邻的所有子节点。

(3) SelectedNode 属性

该属性返回用户从 TreeView 控件中选定的一个 TreeNode 对象。例如，以下语句在标签 Label1 中显示选择节点的文本：

```
Label1.Text = "选择的节点是:" + TreeView1.SelectedNode.Text;
```

(4) Nodes 属性

Nodes 属性是 TreeView 控件中所有节点的集合，是 TreeNodeCollection 类对象。Nodes 集合属性中的每一个节点都是一个 TreeNode 对象。该集合属性的使用方法与 7.4 节介绍的 DropDownList 控件的 Items 属性相似。

可以通过索引来表示 Nodes 集合中的元素(索引从零开始)，但要注意节点的层次结构。例如：

TreeView1.Nodes 表示 TreeView1 控件的所有节点集合。

TreeView1.Nodes[0]表示 TreeView1 控件中第一个根节点。

TreeView1.Nodes[0].ChildNodes 表示 TreeView1 控件中第一个根节点的子节点集合。

TreeView1.Nodes[0].ChildNodes[1]表示 TreeView1 控件中第一个根节点的第 2 个子节点。

TreeView 控件的常用方法及其说明如表 9.5 所示。

表 9.5　TreeView 控件的常用方法及其说明

方　　法	说　　明
ExpandAll	打开树中的每个节点
FindNode	检索 TreeView 控件中指定值路径处的 TreeNode 对象

TreeView 控件的常用事件及其说明如表 9.6 所示。

表 9.6　TreeView 控件的常用事件及其说明

事　　件	说　　明
SelectedNodeChanged	当选择 TreeView 控件中的节点时发生
TreeNodeCheckChanged	当 TreeView 控件中的复选框在向服务器的两次发送过程之间状态有所更改时发生
TreeNodeCollapsed	当折叠 TreeView 控件中的节点时发生
TreeNodeDataBound	当数据项绑定到 TreeView 控件中的节点时发生
TreeNodeExpanded	当扩展 TreeView 控件中的节点时发生
TreeNodePopulate	当其 PopulateOnDemand 属性设置为 True 的节点在 TreeView 控件中展开时发生

9.3.5　TreeView 控件的应用

1. 通过 DataSourceID 属性设置 TreeView 控件的数据源

ASP .NET 提供了 SiteMapDataSource 和 XmlDataSource 两个服务器控件，位于工具箱的“数据”选项卡中，用于 ASP .NET 站点导航，前者检索站点地图提供程序的导航数据；后者检索指定的 XML 文件的导航数据，并将导航数据传递到可显示该数据的控件（如 TreeView 和 Menu 控件）。

【练一练】 在本章网站 CH9 中添加一个 webform3 网页，其功能是说明通过 DataSourceID 属性设置 TreeView 控件的数据源来实现网点导航的使用方法。

其设计步骤如下：

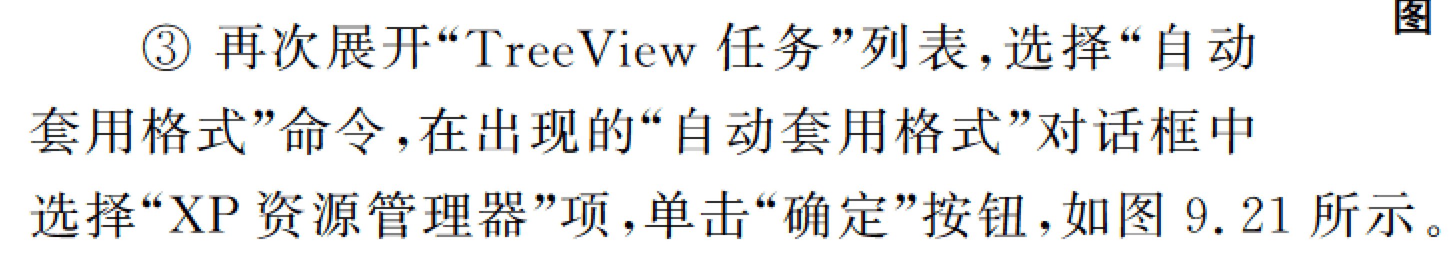
① 打开 CH9 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH9”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform3. aspx，其他保持默认项，单击“添加”按钮。

② 从工具箱的“导航”类别中拖曳一个 TreeView 控件 TreeView1 到网页中，展开 TreeView 任务列表，如图 9.19 所示，选择“新建数据源”命令，在出现的“数据源配置向导”对话框中选择“站点地图”项，单击“确定”按钮，如图 9.20 所示。



图 9.19　展开“TreeView 任务”列表

这样就创建了 SiteMapDataSource1 控件，它对应前面创建的站点地图 web. sitemap 文件。



③ 再次展开“TreeView 任务”列表，选择“自动套用格式”命令，在出现的“自动套用格式”对话框中选择“XP 资源管理器”项，单击“确定”按钮，如图 9.21 所示。

④ 进入源视图，将 TreeView1 控件的 Font-Size 属性改为 14pt。webform3 网页的最终设计界面如图 9.22 所示。

⑤ 在 CH9 网站中添加一个 disp. aspx 的网页，其中只有一个 Label1 标签控件，它的源视图代码如下：

```
<asp:Label ID = "Label1" runat = "server" style = "color: #FF00FF; font - size: 24px;
    font - weight: 700; font - family: 仿宋" />
```

⑥ 单击工具栏中的 ▶ Internet Explorer 按钮执行 webform3 网页，显示的站点地图中的每一项都是一个超链接，例如单击“计算机学院”项，在新的窗口中打开 disp 网页，其结果如图 9.23 所示。



图 9.20　选择“站点地图”项



图 9.21　选择“XP 资源管理器”项

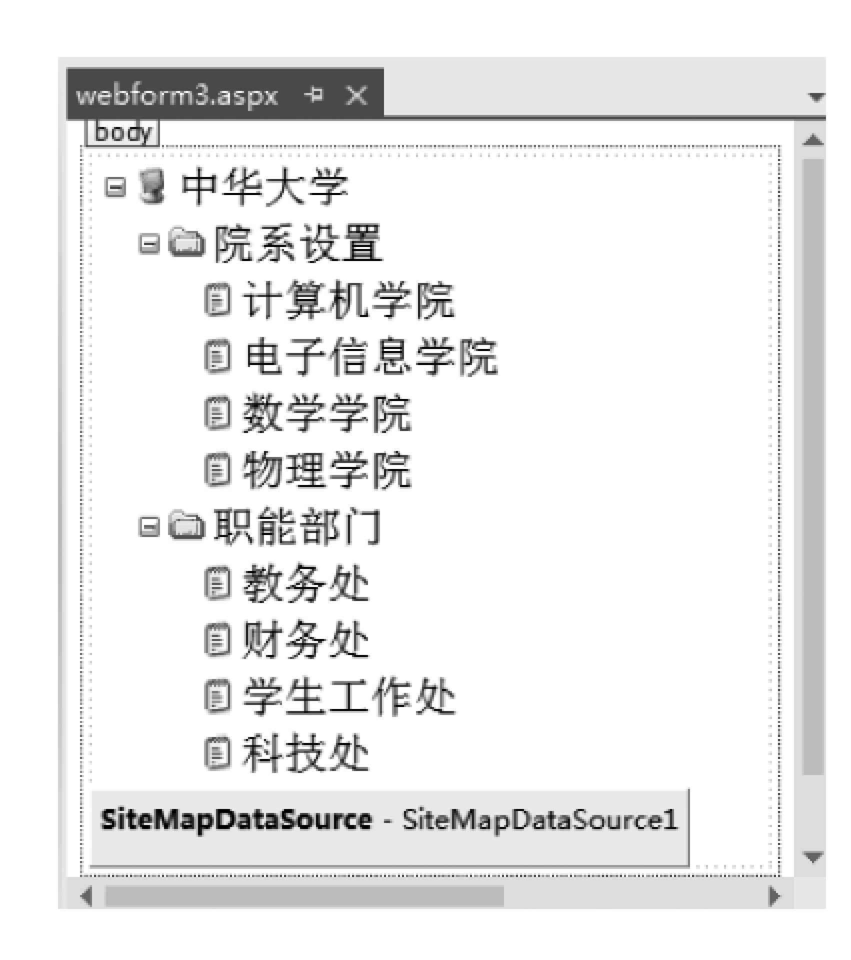


图 9.22　webform3 网页的最终设计界面





图 9.23　webform3 网页的执行过程

2. 向 TreeView 控件中添加节点的方法来设置 TreeView 控件的数据源

向 TreeView 控件添加节点有以下几种方法。

(1) 手工方式添加节点

向网页中拖曳一个 TreeView1 控件，展开“TreeView 任务”列表，从中选择“编辑节点”命令，打开“TreeView 节点编辑器”对话框，可以从中添加和删除节点，如图 9.24 所示，每个节点至少应设置 Text 和 Vlaue 属性，还可以根据需要设置 NavigateUrl 和 Target 属性等。该图设置的 TreeView1 控件的源视图代码如下：

```
<asp:TreeView ID = "TreeView1" runat = "server">
    <Nodes>
      <asp:TreeNode Text = "中华大学" Value = "中华大学">
        <asp:TreeNode Text = "院系设置" Value = "院系设置">
          <asp:TreeNode Text = "计算机学院" Value = "计算机学院"></asp:TreeNode>
          <asp:TreeNode Text = "电子信息学院" Value = "电子信息学院"></asp:TreeNode>
          <asp:TreeNode Text = "数学学院" Value = "数学学院"></asp:TreeNode>
          <asp:TreeNode Text = "物理学院" Value = "物理学院"></asp:TreeNode>
        </asp:TreeNode>
        <asp:TreeNode Text = "职能部门" Value = "职能部门">
          <asp:TreeNode Text = "教务处" Value = "教务处"></asp:TreeNode>
          <asp:TreeNode Text = "财务处" Value = "财务处"></asp:TreeNode>
          <asp:TreeNode Text = "学生工作处" Value = "学生工作处"></asp:TreeNode>
          <asp:TreeNode Text = "科技处" Value = "科技处"></asp:TreeNode>
        </asp:TreeNode>
      </asp:TreeNode>
    </Nodes>
</asp:TreeView>
```



图 9.24　“TreeView 节点编辑器”对话框

(2) 通过编程方式添加节点

由于 TreeView 控件的 Nodes 属性是一个 TreeNodeCollection 类对象，因此采用 Add 方法向其中添加 TreeNode 对象。这种方式在可以在运行时动态地增加或删除 TreeView 控件的节点。

【练一练】 在本章网站 CH9 中添加一个 webform4 网页，其功能是说明通过 DataSourceID 属性设置 TreeView 控件的数据源来实现网点导航的使用方法。

其设计步骤如下：

① 打开 CH9 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH9(1)”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform4.aspx，其他保持默认项，单击“添加”按钮。

② 在网页中插入一个 1 行 2 列的表格，在第 1 列中放置一个 TreeView 控件 TreeView1，展开“TreeView 任务”列表，选择“自动套用格式”命令，在出现的“自动套用格式”对话框中选择“XP 资源管理器”项，单击“确定”按钮。

③ 进入网页源视图，将 TreeView1 控件的 Font-Size 属性改为 14pt。在表格的第 2 列中用代码插入一个 Iframe1 框架。本网页的源视图代码如下：

```
<%@ Page Language = "C#" AutoEventWireup = "true" CodeFile = "webform4.aspx.cs"
    Inherits = "webform4" %>
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head runat = "server">
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8"/>
    <title></title>
  </head>
  <body>
    <form id = "form1" runat = "server">
      <div>
        <table class = "auto-style1">
          <tr>
            <td style = "width:100px">
              <asp:TreeView ID = "TreeView1" runat = "server"
                ImageSet = "XPFileExplorer" NodeIndent = "15">
                <HoverNodeStyle Font-Underline = "True" ForeColor = "#6666AA" />
                <NodeStyle Font-Names = "Tahoma" Font-Size = "14pt"
                  ForeColor = "Black" HorizontalPadding = "2px" NodeSpacing = "0px"
                  VerticalPadding = "2px" />
                <ParentNodeStyle Font-Bold = "False" />
                <SelectedNodeStyle BackColor = "#B5B5B5" Font-Underline = "False"
                  HorizontalPadding = "0px" VerticalPadding = "0px" />
              </asp:TreeView>
            </td>
            <td>
              <iframe name = "Iframe1" style = " height:300px; width:211px;
                text-align:center" id = "Iframe1" >
              </iframe>
            </td>
          </tr>
        </table>
      </div>
    </form>
```

```
    </body>
</html>
```

④ 在网页上设计如下事件处理方法：

```
protected void Page_Load(object sender, EventArgs e)
{   TreeView1.Nodes.Clear();
    TreeNode node = new TreeNode("中华大学");
    node.Target = "Iframe1";
    node.NavigateUrl = "disp.aspx?mc = 中华大学";
    TreeView1.Nodes.Add(node);
    node = new TreeNode("院系设置");
    node.Target = "Iframe1";
    node.NavigateUrl = "disp.aspx?mc = 院系设置";
    TreeView1.Nodes[0].ChildNodes.Add(node);
    node = new TreeNode("计算机学院");
    node.Target = "Iframe1";
    node.NavigateUrl = "disp.aspx?mc = 计算机学院";
    TreeView1.Nodes[0].ChildNodes[0].ChildNodes.Add(node);
    node = new TreeNode("电子信息学院");
    node.Target = "Iframe1";
    node.NavigateUrl = "disp.aspx?mc = 电子信息学院";
    TreeView1.Nodes[0].ChildNodes[0].ChildNodes.Add(node);
    node = new TreeNode("数学学院");
    node.Target = "Iframe1";
    node.NavigateUrl = "disp.aspx?mc = 数学学院";
    TreeView1.Nodes[0].ChildNodes[0].ChildNodes.Add(node);
    node = new TreeNode("物理学院");
    node.Target = "Iframe1";
    node.NavigateUrl = "disp.aspx?mc = 物理学院";
    TreeView1.Nodes[0].ChildNodes[0].ChildNodes.Add(node);
    node = new TreeNode("职能部门");
    node.Target = "Iframe1";
    node.NavigateUrl = "disp.aspx?mc = 职能部门";
    TreeView1.Nodes[0].ChildNodes.Add(node);
    node = new TreeNode("教务处");
    node.Target = "Iframe1";
    node.NavigateUrl = "disp.aspx?mc = 教务处";
    TreeView1.Nodes[0].ChildNodes[1].ChildNodes.Add(node);
    node = new TreeNode("财务处");
    node.Target = "Iframe1";
    node.NavigateUrl = "disp.aspx?mc = 财务处";
    TreeView1.Nodes[0].ChildNodes[1].ChildNodes.Add(node);
    node = new TreeNode("学生工作处");
    node.Target = "Iframe1";
    node.NavigateUrl = "disp.aspx?mc = 学生工作处";
    TreeView1.Nodes[0].ChildNodes[1].ChildNodes.Add(node);
    node = new TreeNode("科技处");
    node.Target = "Iframe1";
    node.NavigateUrl = "disp.aspx?mc = 科技处";
    TreeView1.Nodes[0].ChildNodes[1].ChildNodes.Add(node);
}
```

上述代码在 TreeView1 控件中添加 11 个节点，并设置各个节点的 Text、Target 和 NavigateUrl 属性。其中要转向的 disp 网页已经在前面例子中创建好了。

⑤ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页,用户单击 TreeView1 控件中的"计算机学院"项,其显示结果如图 9.25 所示。

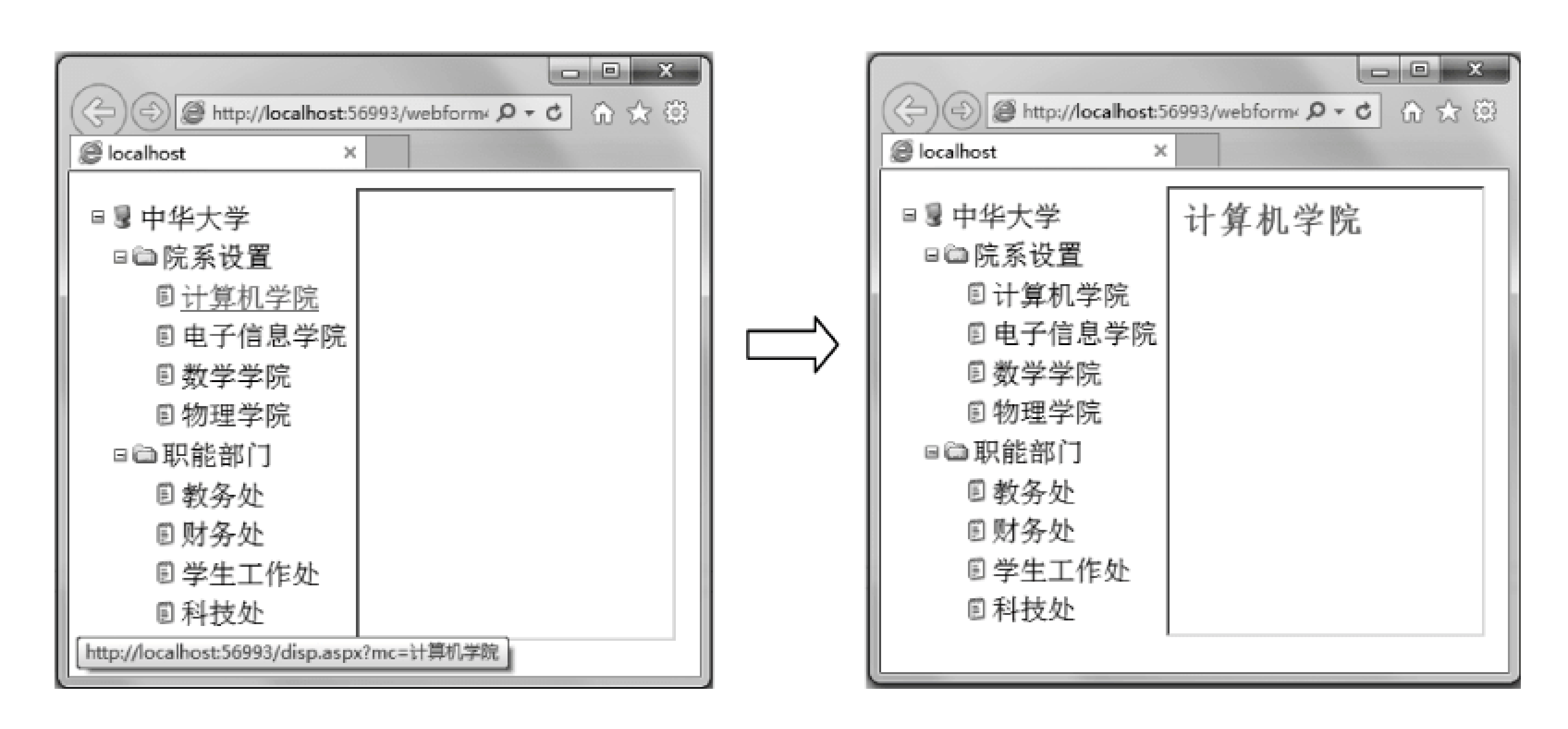


图 9.25 webform4 网页的执行过程

与上例相比,本例可以控制转向的网页在同一页面的框架中呈现,使得网页界面设计更加紧凑。

9.4 练 习 题

1. 单项选择题

(1) 应用程序主题存储于网站的(　　)目录中。

A. App_Themes　B. 根　C. App_Code　D. 任何子目录

(2) 为单个页面指定主题,只需要在页面的(　　)指令中进行设置。

A. Lanuage　B. @ Page　C. @ include　D. @ code

(3) 若要对特定网页取消主题设置,可以将其页面指令@ Page 的 Theme 属性设置为(　　)。

A. True　B. 空字符串　C. False　D. This

(4) 以下关于主题的叙述中错误的是(　　)。

A. 一个网页只能应用一个主题,但该主题中可以有多个外观文件

B. 主题是由外观、级联样式表、图像和其他资源组成

C. 不能通过 web.config 文件为应用程序中的所有页定义应用的主题

D. 一个主题可以用于同一网站的多个网页

(5) 控件属性的应用顺序是(　　)。

① 应用 StyleSheetTheme 属性。

② 应用网页中的控件属性。

③ 应用 Theme 属性。

A. ①②③　B. ①③②　C. ③①②　D. ②③①

(6) 母版页文件的扩展名是(　　)。

A. config　B. master　C. asp　D. aspx

(7) 以下关于母版页的叙述中错误的是(　　)。

A. 母版页技术由(母版页本身)和(内容页)两部分组成

B. 在内容页中设定 MasterPageFile 属性以指定所使用的(@ Page 指令)
C. Content 是内容页的内容和控件的容器,与母版页上的 ContentPlaceHolder 控件相对应
D. 一个内容页可以使用多个母版页

(8) 以下关于母版页的叙述中正确的是(　　)。
A. 一个母版页也可以有母版页
B. 一个母版页只能用于一个内容页
C. 一个网站只能有一个母版页
D. 一个内容页可以作为另一个内容页的母版页

(9) 以下关于母版页的叙述中正确的是(　　)。
A. 母版页不能应用主题
B. 母版页能在浏览器上执行
C. 内容页可以有<html>、<head>、<body>和执行在服务器端的<form>标记
C. 母版页不能有<html>、<head>、<body>和执行在服务器端的<form>标记

(10) ContentPlaceHolder 控件即是(　　)。
A. 不变区域　　B. 母版标记　　C. 内容占位符　　D. 内容标记

(11) Content 控件的 ContentPlaceHolderID 一定要与母版页中 ContentPlaceHolder 控件的(　　)属性值对应。
A. ID　　B. Inherits　　C. Style　　D. font

(12) 以下关于站点地图的叙述中正确的是(　　)。
A. 站点地图的文件名为 Web. Sitemap,其本质是一个普通的 HTML 文件
B. 站点地图的文件名只能是 Web. Sitemap,不能是其他文件名
C. 站点地图不能用其他文本编辑器直接创建和修改
D. 只有名称为 Web. sitemap 的站点地图才会被自动加载,并且必须出现在网站的根目录中

(13) TreeView 控件的节点集合保存在(　　)属性中。
A. Items　　B. Nodes　　C. Controls　　D. ImageList

(14) treeView1. Nodes[0]. Nodes[1]代表了控件 treeView1 的(　　)。
A. 第 1 个根节点的第 1 个子节点　　B. 第 1 个根节点的第 2 个子节点
C. 第 2 个根节点的第 1 个子节点　　D. 第 2 个根节点的第 2 个子节点

(15) 如果要设置 TreeView 控件的数据源,要使用(　　)属性。
A. DataSource　　B. DataSourceID　　C. TreeView　　D. TreeNode

(16) TreeView 控件中节点的类型是(　　)。
A. DataSource　　B. DataSourceID　　C. TreeView　　D. TreeNode

2. 问答题

(1) 简述主题的作用。
(2) 简述一个网页应用主题的几种方式。
(3) 简述母版页的运行过程。
(4) 简述母版页的优点。
(5) 简述 TreeView 控件的作用。

(6) 简述 Web. sitmap 文件、SiteMapDataSource 和 TreeView 控件的关系。

9.5　上机实验题

在 CH9 网站中添加一个 Exp 网页，其设计界面如图 9.26 所示，网页中有一个 student 类，建立 st 数组如下：

```
student[] st = new student[5] { new student(1,"王华","女","汉族","15001"),
     new student(2,"孙丽","女","满族","15002"),
     new student(3,"李兵","男","汉族","15001"),
     new student(6,"张军","男","汉族","15001"),
     new student(8,"马棋","男","回族","15002") };
```

用户单击左边的 TreeView1 控件中的节点时，在右边的 ListBox1 控件中显示满足条件的学生记录。图 9.27 所示是用户单击"男"节点的结果。

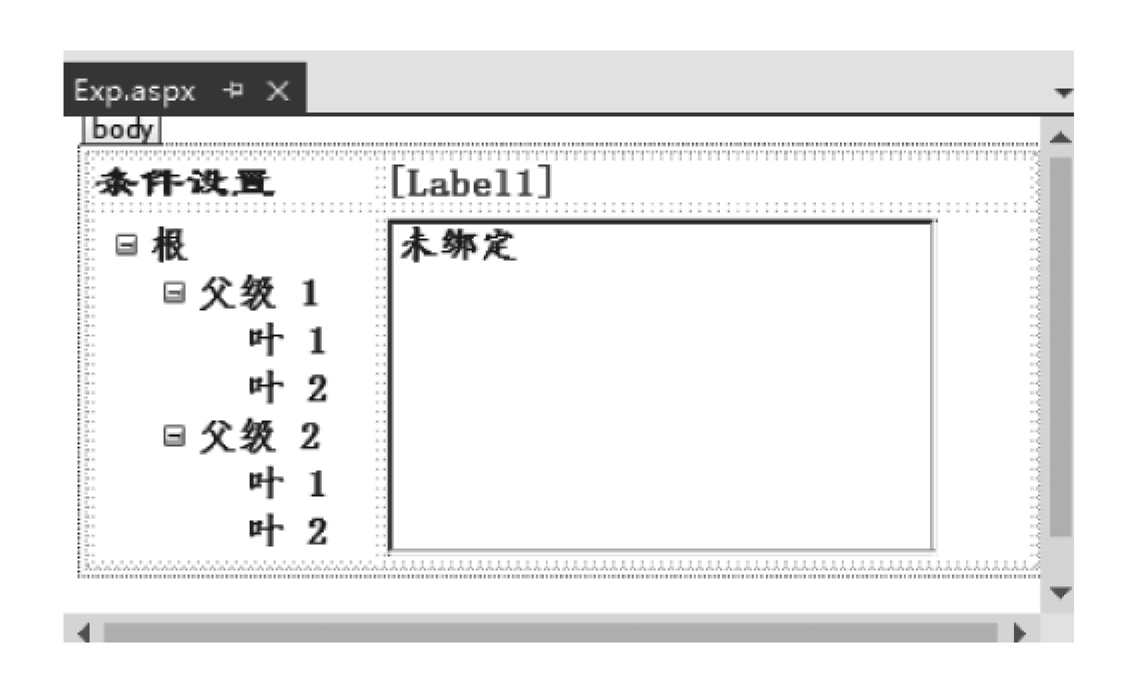


图 9.26　上机实验题网页的设计界面

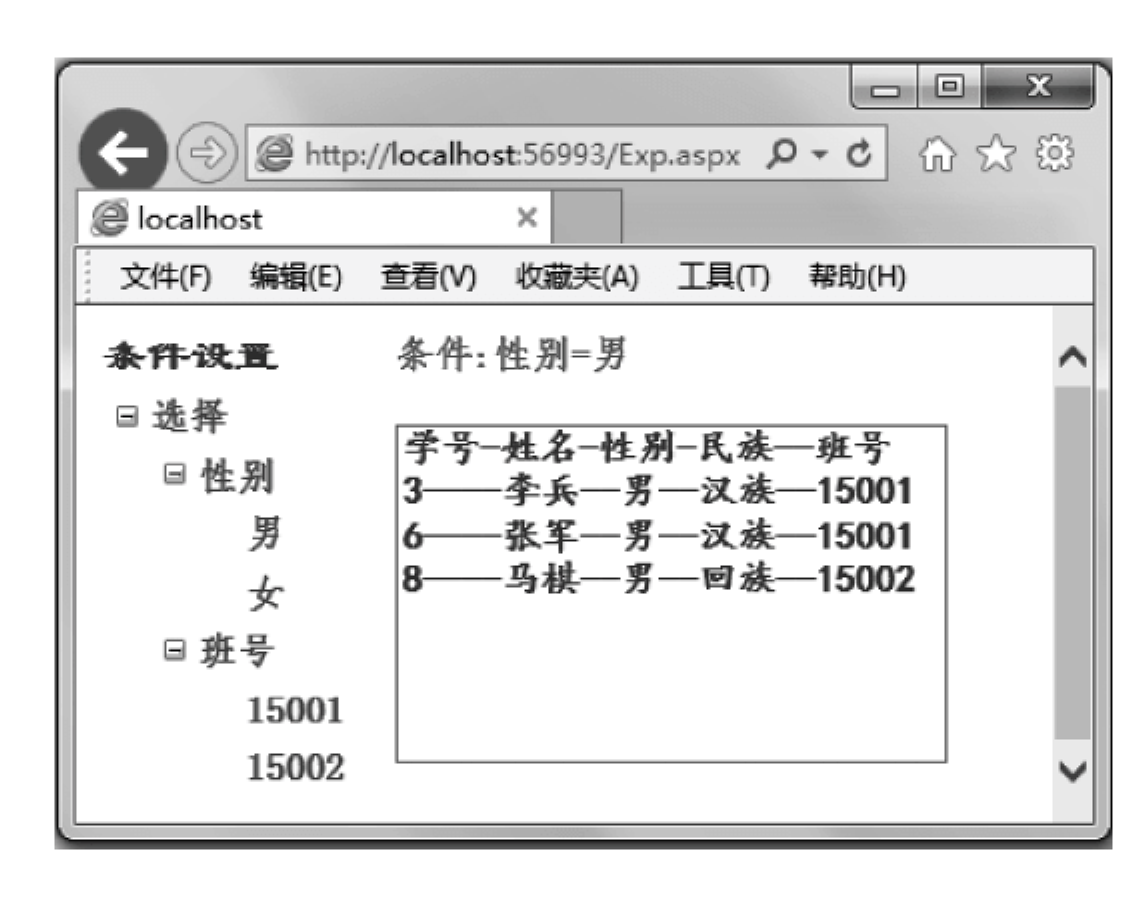
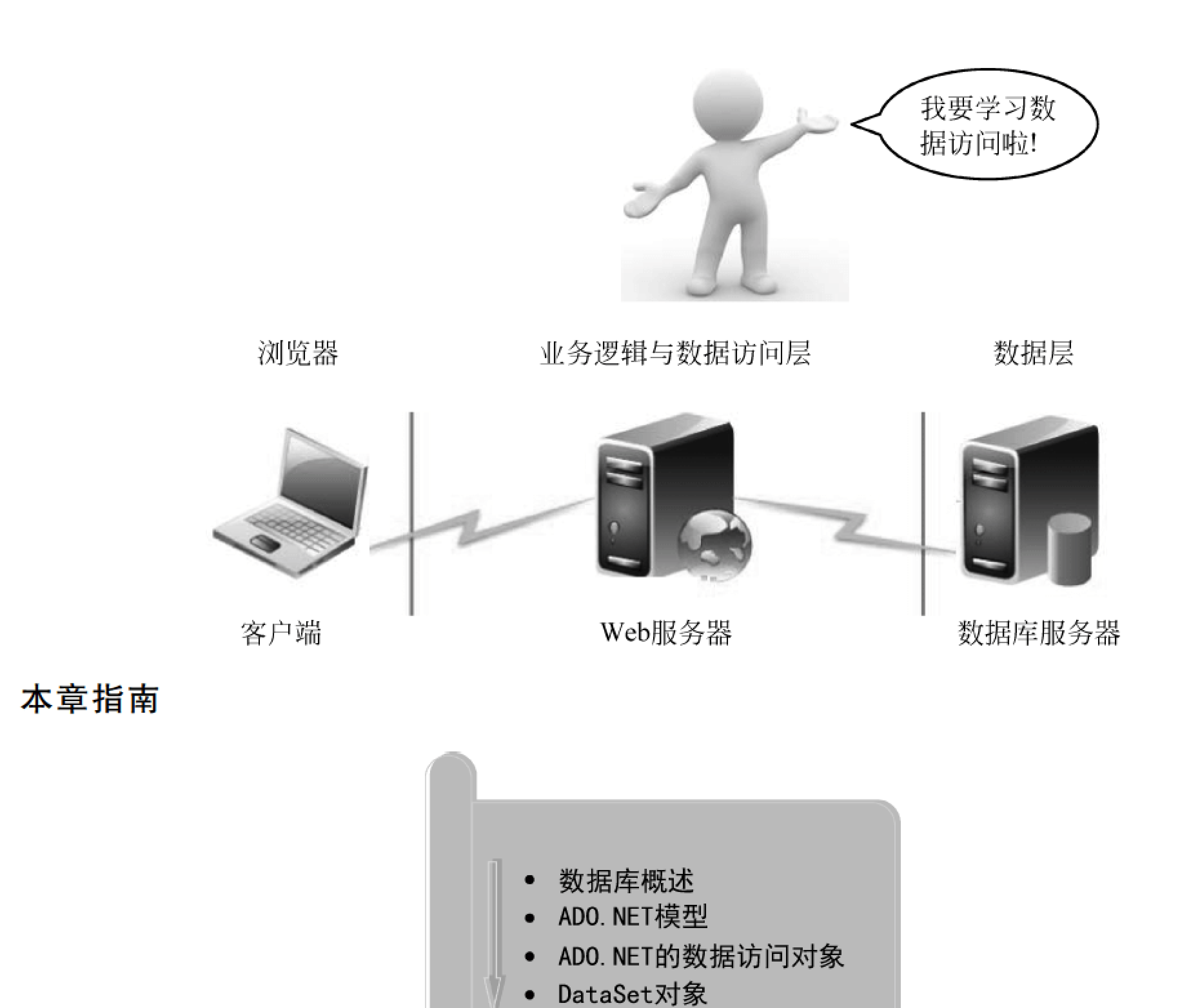


图 9.27　上机实验题网页的执行界面

第 10 章　ASP .NET 数据库编程

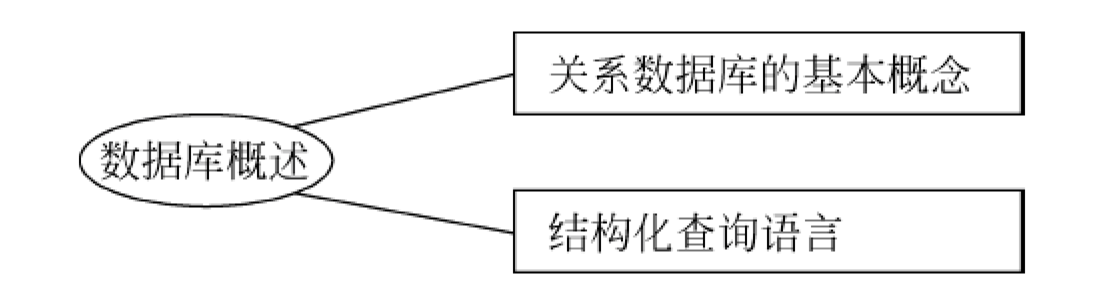


本章指南

- 数据库概述
- ADO.NET模型
- ADO.NET的数据访问对象
- DataSet对象

10.1　数据库概述

知识梳理



10.1.1　关系数据库的基本概念

数据库用于存储结构化数据。数据组织有多种数据模型，目前主要的数据模型是关系数据模型，以关系模型为基础的数据库就是关系数据库。

关系数据库以表的形式(即关系)组织数据。关系数据库以关系的数学理论为基础。在关

系数据库中，用户可以不必关心数据的存储结构，同时，关系数据库的查询可用高级语言来描述，这大大提高了查询效率。下面讨论关系数据库的基本术语。

- 表：表用于存储数据，它以行列方式组织，可以使用 SQL 从中获取、修改和删除数据。表是关系数据库的基本元素。表在现实生活中随处可见，如职工表、学生表和统计表等。表具有直观、方便和简单的特点。
- 记录：记录是指表中的一行，在一般情况下，记录和行的意思是相同的。
- 字段：字段是表中的一列，在一般情况下，字段和列所指的内容是相同的。
- 关系：关系是一个从数学中来的概念，在关系代数中，关系是指二维表，表既可以用来表示数据，也可以用来表示数据之间的联系。
- 索引：索引是建立在表上的单独的物理数据库结构，基于索引的查询使数据获取更为快捷。索引是表中的一个或多个字段，索引可以是唯一的，也可以是不唯一的，主要是看这些字段是否允许重复。主索引是表中的一列和多列的组合，作为表中记录的唯一标识。外部索引是相关联的表的一列或多列的组合，通过这种方式来建立多个表之间的联系。
- 视图：视图是一个与真实表相同的虚拟表，用于限制用户可以看到和修改的数据量，从而简化数据的表达。
- 存储过程：存储过程是一个编译过的 SQL 程序。在该过程中可以嵌入条件逻辑、传递参数、定义变量和执行其他编程任务。

10.1.2　结构化查询语言

结构化查询语言(SQL)是目前各种关系数据库管理系统广泛采用的数据库语言，很多数据库和软件系统都支持 SQL 或提供 SQL 语言接口。

注意：本章和第 11 章以第 2 章创建的 school 数据库及其数据表为例讨论 ASP .NET 数据库编程。

1. SQL 语言的组成

SQL 语言包含查询、操纵、定义和控制等几个部分。它们都是通过命令动词分开的，各种语句类型对应的命令动词如下：

- 数据查询的命令动词为 SELECT；
- 数据定义的命令动词为 CREATE、DROP；
- 数据操纵的命令动词为 INSERT、UPDATE、DELETE；
- 数据控制的命令动词为 GRANT、REVOKE。

2. 数据定义语言

(1) CREATE 语句

CREATE 语句用于建立数据表，其基本格式如下：

```
CREATE TABLE 表名
(列名 1 数据类型 1 [NULL | NOT NULL] [PRIMARY KEY | UNIQUE]
[,列名 2 数据类型 2 [NOT NULL]]…)
```

例如，创建 school 数据库中 student 表对应的 SQL 语句如下：

```
CREATE TABLE student
```

```
( 学号 int NOT NULL PRIMARY KEY,      /* 学号字段为主键 */ /
  姓名 CHAR(10),
  性别 ChAR(2),
  民族 CHAR(10),
  班号 CHAR(10))
```

(2) DROP 语句

DROP 语句用于删除数据表,其基本格式如下:

```
DROP TABLE 表名
```

3. 数据操纵语言

(1) INSERT 语句

INSERT 语句用于在一个表中添加新记录,然后给新记录的字段赋值。其基本格式如下:

```
INSERT INTO 表名[(列名 1[,列名 2, … ])]
VALUES(表达式 1[,表达式 2, … ])
```

(2) UPDATE 语句

UPDATE 语句用于新的值更新表中的记录。其基本格式如下:

```
UPDATE 表名
    SET 列名 1 = 表达式 1
    [,SET 列名 2 = 表达式 2] …
WHERE 条件表达式
```

(3) DELETE 语句

DELETE 语句用于删除记录,其基本格式如下:

```
DELETE FROM 表名
[WHERE 条件表达式]
```

4. 数据查询语句

SQL 的数据查询语句是使用很频繁的语句。SELECT 的基本格式如下:

```
SELECT 字段表
FORM 表名
WHERE 查询条件
GROUP BY 分组字段
HAVING 分组条件
ORDER BY 字段[ASC|DESC]
```

各子句的功能如下:

- SELECT:指定要查询的内容。
- FORM:指定从其中选定记录的表名。
- WHERE:指定所选记录必须满足的条件。
- GROUP BY:把选定的记录分成特定的组。
- HAVING:说明每个组需要满足的条件。
- ORDER BY:按特定的次序将记录排序。

其中,在"字段表"中可使用聚合函数对记录进行合计,它返回一组记录的单一值,常见的聚合函数如表 10.1 所示。"查询条件"由常量、字段名、逻辑运算符、关系运算符等组成,其中

的关系运算符如表 10.2 所示。

表 10.1　常见的聚合函数

函数名	说　明	函数名	说　明
AVG	返回特定字段中值的平均数	MAX	返回特定字段中的最大值
COUNT	返回选定记录的个数	MIN	返回特定字段中的最小值
SUM	返回特定字段中所有值的总和		

表 10.2　关系运算符

符　号	说　明	符　号	说　明
<	小于	<=	小于等于
>	大于	>=	大于等于
=	等于	<>	不等于
IN (一组值)	在一组值中	LIKE*	与一个通配符匹配
BETWEEN 值 1 AND 值 2	在两数值之间		

* 通配符可使用“?”代表一个字符位，“%”代表零个或多个字符位。

【练一练】 以 school 数据库为例，设计满足如下功能的 SQL 语句：

(1) 查询 student 表中 15002 班所有学生记录。

(2) 查询所有学生的学号、姓名、课程名和分数。

(3) 查询所有学生的学号、姓名、课程名和分数，要求按课程名排序。

(4) 查询分数在 80～90 之间的所有学生的学号、姓名、课程名和分数，并按分数递减排列。

(5) 查询每个班每门课程的平均分。

(6) 查询最高分的学生姓名和班号。

其设计步骤如下：

① 启动 SQL Server 2012。

② 单击工具栏中的「新建查询(N)」按钮，在出现的编辑窗口中输入实现功能(1)的 SQL 语句：

```
USE school                                      /* 指定 school 数据库 */
SELECT * FROM student WHERE 班号 = '15002'
```

单击「! 执行(X)」按钮，其执行结果如图 10.1 所示。

③ 实现功能(2)的 SQL 语句：

```
USE school
SELECT student.学号,student.姓名,course.课程名,score.分数
FROM student,course,score
WHERE student.学号 = score.学号 AND course.课程号 = score.课程号
```

其执行结果如图 10.2 所示。

④ 实现功能(3)的 SQL 语句：

```
USE school
SELECT student.学号,student.姓名,course.课程名,score.分数
FROM student,course,score
WHERE student.学号 = score.学号 AND course.课程号 = score.课程号
ORDER BY course.课程名
```

结果　消息

	学号	姓名	性别	民族	班号
1	2	孙丽	女	满族	15002
2	8	马棋	男	回族	15002

图 10.1　功能(1)的执行结果

结果　消息

	学号	姓名	课程名	分数
1	1	王华	C语言	80
2	1	王华	数据结构	83
3	2	孙丽	C语言	70
4	2	孙丽	数据结构	52
5	3	李兵	C语言	76
6	3	李兵	数据结构	70
7	6	张军	C语言	90
8	6	张军	数据结构	92
9	8	马棋	C语言	88
10	8	马棋	数据结构	79

图 10.2　功能(2)的执行结果

其执行结果如图 10.3 所示。

⑤ 实现功能(4)的 SQL 语句：

```
USE school
SELECT student.学号,student.姓名,course.课程名,score.分数
FROM student,course,score
WHERE student.学号 = score.学号 AND course.课程号 = score.课程号
    AND student.学号 = score.学号 AND score.分数 BETWEEN 80 AND 90
ORDER BY score.分数 DESC
```

其执行结果如图 10.4 所示。

结果　消息

	学号	姓名	课程名	分数
1	1	王华	C语言	80
2	2	孙丽	C语言	70
3	3	李兵	C语言	76
4	6	张军	C语言	90
5	8	马棋	C语言	88
6	1	王华	数据结构	83
7	2	孙丽	数据结构	52
8	3	李兵	数据结构	70
9	6	张军	数据结构	92
10	8	马棋	数据结构	79

图 10.3　功能(3)的执行结果



结果　消息

	学号	姓名	课程名	分数
1	6	张军	C语言	90
2	8	马棋	C语言	88
3	1	王华	数据结构	83
4	1	王华	C语言	80

图 10.4　功能(4)的执行结果

⑥ 实现功能(5)的 SQL 语句：

```
USE school
SELECT student.班号,course.课程名,AVG(score.分数) AS '平均分'
FROM student,course,score
WHERE student.学号 = score.学号 AND course.课程号 = score.课程号
GROUP BY student.班号,course.课程名
```

其执行结果如图 10.5 所示。

⑦ 实现功能(6)的 SQL 语句：

```
SELECT student.姓名,student.班号,course.课程名,score.分数
USE school
FROM student,course,score
WHERE student.学号 = score.学号 AND course.课程号 = score.课程号
```

```
AND score.分数 =
(SELECT MAX(分数) FROM score)
```

其执行结果如图 10.6 所示。

	班号	课程名	平均分
1	15001	C语言	82
2	15002	C语言	79
3	15001	数据结构	81.6666666666667
4	15002	数据结构	65.5

图 10.5　功能(5)的执行结果

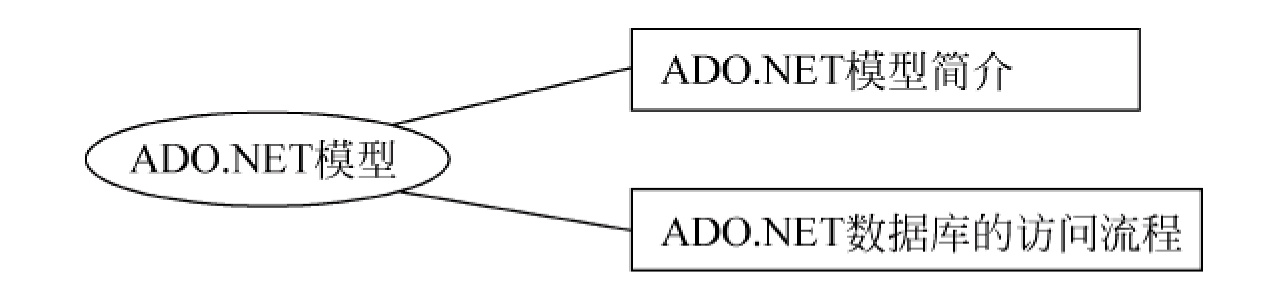


	姓名	班号	课程名	分数
1	张军	15001	数据结构	92

图 10.6　功能(6)的执行结果

10.2　ADO .NET 模型

知识梳理



10.2.1　ADO .NET 模型简介

在 SQL Server 中访问自己所建立的数据库是十分方便的,但是如何从外部访问 SQL Server 数据库呢? 为此人们提出了各种数据库的访问模型,ADO .NET 是微软公司的新一代 .NET 数据库访问模型,它是目前数据库程序设计师用来开发数据库应用程序的主要接口。

ADO .NET 是在 .NET Framework 上访问数据库的一组类库,它利用 .NET Data Provider(数据提供程序)以进行数据库的连接与访问,通过 ADO .NET,数据库程序设计人员能够很轻易地使用各种对象来访问符合自己需求的数据库内容。换句话说,ADO .NET 定义了一个数据库访问的标准接口,提供数据库管理系统的各个厂商可以根据此标准开发对应的 .NET Data Provider,这样编写数据库应用程序的人员不必了解各类数据库底层运作的细节,只要学会 ADO .NET 所提供对象的模型,便可轻易地访问所有支持 .NET Data Provider 的数据库。

ADO .NET 是应用程序和数据源之间沟通的桥梁。通过 ADO .NET 所提供的对象,再配合 SQL 语句就可以访问数据库中的数据,而且凡是能通过 ODBC 或 OLEDB 接口访问的数据库(如 dBase、FoxPro、Excel、Access、SQL Server 和 Oracle 等),也可通过 ADO .NET 来访问。

ADO .NET 主要希望在处理数据的同时,不要一直和数据库联机,而发生一直占用系统资源的现象。为了解决此问题,ADO .NET 将访问数据和数据处理的部分分开,以达到离线访问数据的目的,使得数据库能够运行其他工作。

ADO .NET 模型分成 .NET Data Provider(数据提供程序)和 DataSet 数据集(数据处理的核心)两大主要部分,其中包含的主要组件及其关系如图 10.7 所示。

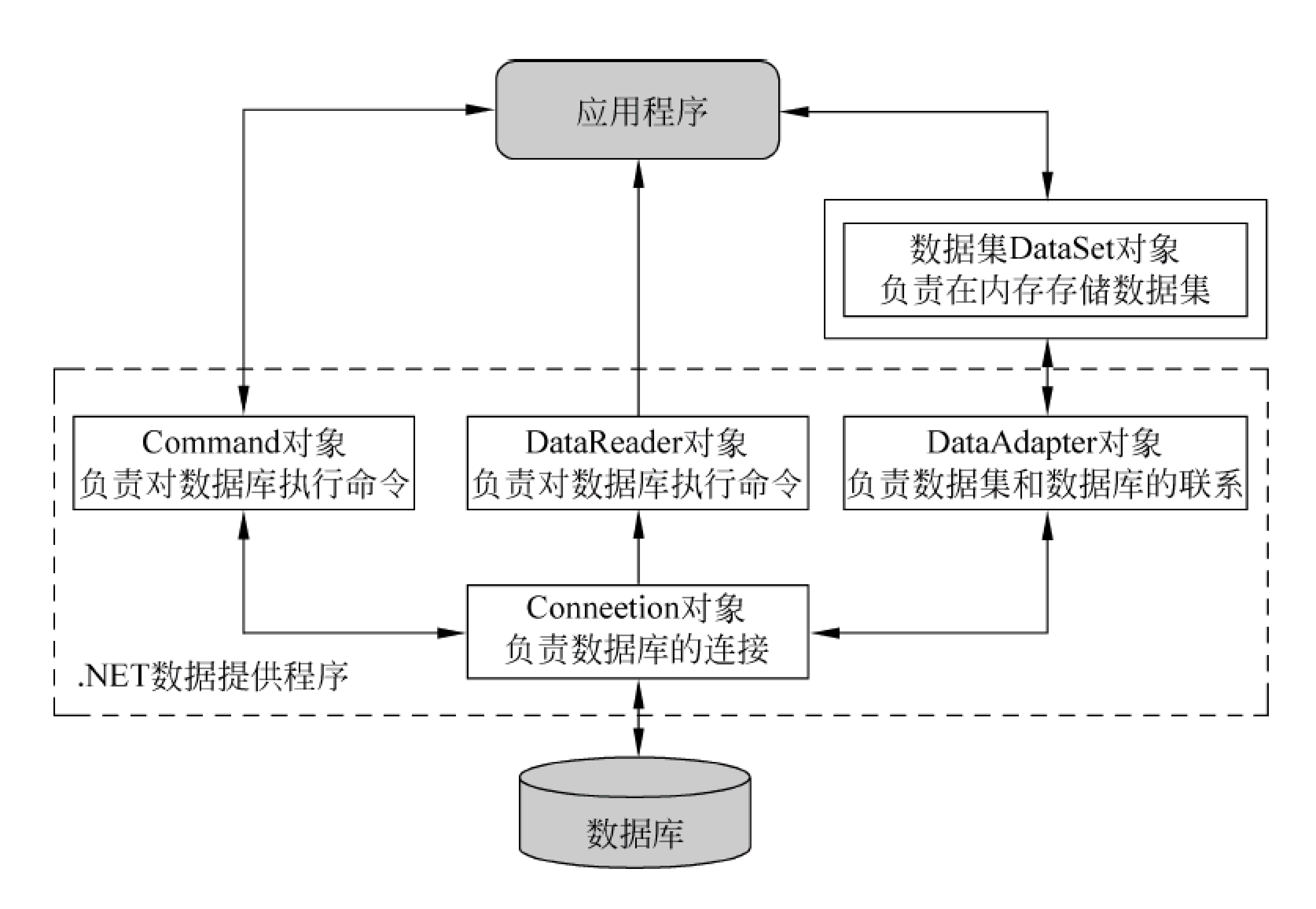


图 10.7　ADO .NET 组件结构模型

1. .NET Data Provider

.NET Data Provider 用于连接到数据库、执行命令和检索结果。可以直接处理检索到的结果，或将其放入 ADO .NET DataSet 对象，以便与来自多个源的数据或在层之间进行远程处理的数据组合在一起，以特殊方式向用户公开。.NET Framework 数据提供程序是轻量的，它在数据源和代码之间创建了一个最小层，以便在不以功能为代价的前提下提高性能。

表 10.3 给出了 .NET Data Provider 中包含的 4 个核心对象。

表 10.3　.NET Data Provider 中包含的 4 个核心对象及其说明

对象	说　明
Connection	提供和数据源的连接功能
Command	提供运行访问数据库命令，传送数据或修改数据的功能，例如运行 SQL 命令和存储过程等
DataAdapter	是 DataSet 对象和数据源间的桥梁。DataAdapter 使用 4 个 Command 对象来运行查询、新建、修改、删除的 SQL 命令，把数据加载到 DataSet，或把 DataSet 内的数据送回数据源
DataReader	通过 Command 对象运行 SQL 查询命令取得数据流，以便进行高速、只读的数据浏览

通过 Connection 对象可与指定的数据库进行连接；Command 对象用来运行相关的 SQL 命令(如 SELECT、INSERT、UPDATE 或 DELETE)，以读取或修改数据库中的数据。通过 DataAdapter 对象中所提供的 4 个 Command 对象来进行离线式的数据访问，这 4 个 Command 对象分别为 SelectCommand、InsertCommand、UpdateCommand 和 DeleteCommand，其中 SelectCommand 是用来将数据库中的数据读出并放到 DataSet 对象中，以便进行离线式的数据访问，至于其他 3 个命令对象(InsertCommand、UpdateCommand 和 DeleteCommand)则是用来修改 DataSet 中的数据，并写回数据库中；通过 DataAdapter 对象的 Fill 方法可以将数据读到 DataSet 中；通过 Update 方法则可以将 DataSet 对象的数据更新到指定的数据库中。

在使用程序管理数据库之前，要先确定使用哪个 Data Provider(数据提供程序)来访问数据库，Data Provider 是一组用来访问数据库的对象，在 .NET Framework 中常用的有如下 4 组数据提供程序：

- SQL .NET Data Provider：支持 SQL Server 7.0 及以上版本，由于它使用自己的通信协议并且做过最优化，所以可以直接访问 SQL Server 数据库，而不必使用 OLEDB 或 ODBC（开放式数据库连接层）接口，因此效果较佳。若程序中使用 SQL .NET Data Provider，则该 ADO .NET 对象名称之前都要加上 Sql，如 SqlConnection、SqlCommand、SqlDataReader 和 SqlDataAdapter 等。在所有使用 SQL .NET Data Provider 的网页中，其引用部分应添加 using System. Data. SqlClient 语句。
- OLEDB .NET Data Provider：支持通过 OLEDB 接口来访问如 dBase、FoxPro、Excel 和 Access 等各类型数据源。程序中若使用 OLEDB .NET Data Provider，则 ADO .NET 对象名称之前要加上 OleDb，如 OleDbConnection、OleDbCommand、OleDbDataReader 和 OleDbDataAdapter 等。在所有使用 OLEDB .NET Data Provider 的网页中，其引用部分应添加 using System. Data. OleDb 语句。
- ODBC .NET Data Provider：支持通过 ODBC 接口来访问如 dBase、FoxPro、Excel、Access、Oracle 以及 SQL Server 等各类型数据源。程序中若使用 ODBC .NET Data Provider，则 ADO .NET 对象名称之前要加上 Odbc，如 OdbcConnection、OdbcCommand、OdbcDataReader 和 OdbcDataAdapter 等。在所有使用 OLEDB .NET Data Provider 的网页中，其引用部分应添加 using System. Data. Odbc 语句。
- ORACLE .NET Data Provider：支持通过 ORACLE 接口来访问 ORACLE 数据源。程序中若使用 ORACLE .NET Data Provider，则 ADO .NET 对象名称之前要加上 Oracle，如 OracleConnection、OracleCommand、OracleDataReader 和 OracleDataAdapter 等。在所有使用 OLEDB .NET Data Provider 的网页中，其引用部分应添加 using System. Data. OracleClient 语句。

从以上介绍看到，访问 SQL Server 数据库，可以使用 SQL .NET Data Provider 和 ODBC .NET Data Provider，但前者可以直接访问 SQL Server 数据库，若使用后者，还需建立 SQL Server 数据库对应的 ODBC 数据源。本书主要介绍使用 SQL .NET Data Provider 直接访问 SQL Server 数据库的方法。

2. DataSet

DataSet（数据集）是 ADO .NET 离线数据访问模型中的核心对象，主要使用时机是在内存中暂存并处理各种从数据源中所取回的数据。DataSet 其实就是一个存放在内存中的数据暂存区，这些数据必须通过 DataAdapter 对象与数据库进行数据交换。在 DataSet 内部允许同时存放一个或多个不同的数据表（Data Table）对象。这些数据表是由数据列和数据域所组成的，并包含有主索引键、外部索引键、数据表间的关系信息以及数据格式的条件限制。

DataSet 的作用像内存中的数据库管理系统，因此在离线时，DataSet 也能独自完成数据的新建、修改、删除、查询等操作，而不必一直局限在和数据库联机时才能做数据维护的工作。DataSet 可以用于访问多个不同的数据源、XML 数据或作为应用程序暂存系统状态的暂存区。

数据库通过 Connection 对象连接后，便可以通过 Command 对象将 SQL 语法（如 INSERT、UPDATE、DELETE 或 SELECT）交由数据库引擎（如 SQL Server）运行，并通过 DataAdapter 对象将数据查询的结果存放到离线的 DataSet 对象中，进行离线数据修改，对降低数据库联机负担具有极大的帮助。至于数据查询部分，还通过 Command 对象设置 SELECT 查询语法和 Connection 对象设置数据库连接，运行数据查询后利用 DataReader 对

象,以只读的方式逐步从前向后浏览记录。

10.2.2　ADO .NET 数据库的访问流程

ADO .NET 访问 SQL Server 数据库的一般流程如下:

① 建立 SqlConnection 对象,创建一个数据库连接。

② 在建立连接的基础上可以使用 SqlCommand 对象对数据库发送查询、新增、修改和删除等命令。

③ 如果只需要一行一行地读记录,创建 SqlDataReader 对象,然后读出记录显示在网页相关控件中,转步骤⑥;否则转步骤④。

④ 创建 SqlDataAdapter 对象,从数据库中取得数据。

⑤ 创建 DataSet 对象,将 SqlDataAdapter 对象填充到 DataSet 对象(数据集)中。然后对 DataSet 对象中的数据进行相应处理,将结果显示在网页相关控件中。

⑥ 关闭数据库。

上述过程如图 10.8 所示。因为一个 DataSet 对象可以容纳多个数据集,如果需要,上述部分过程可以重复操作。

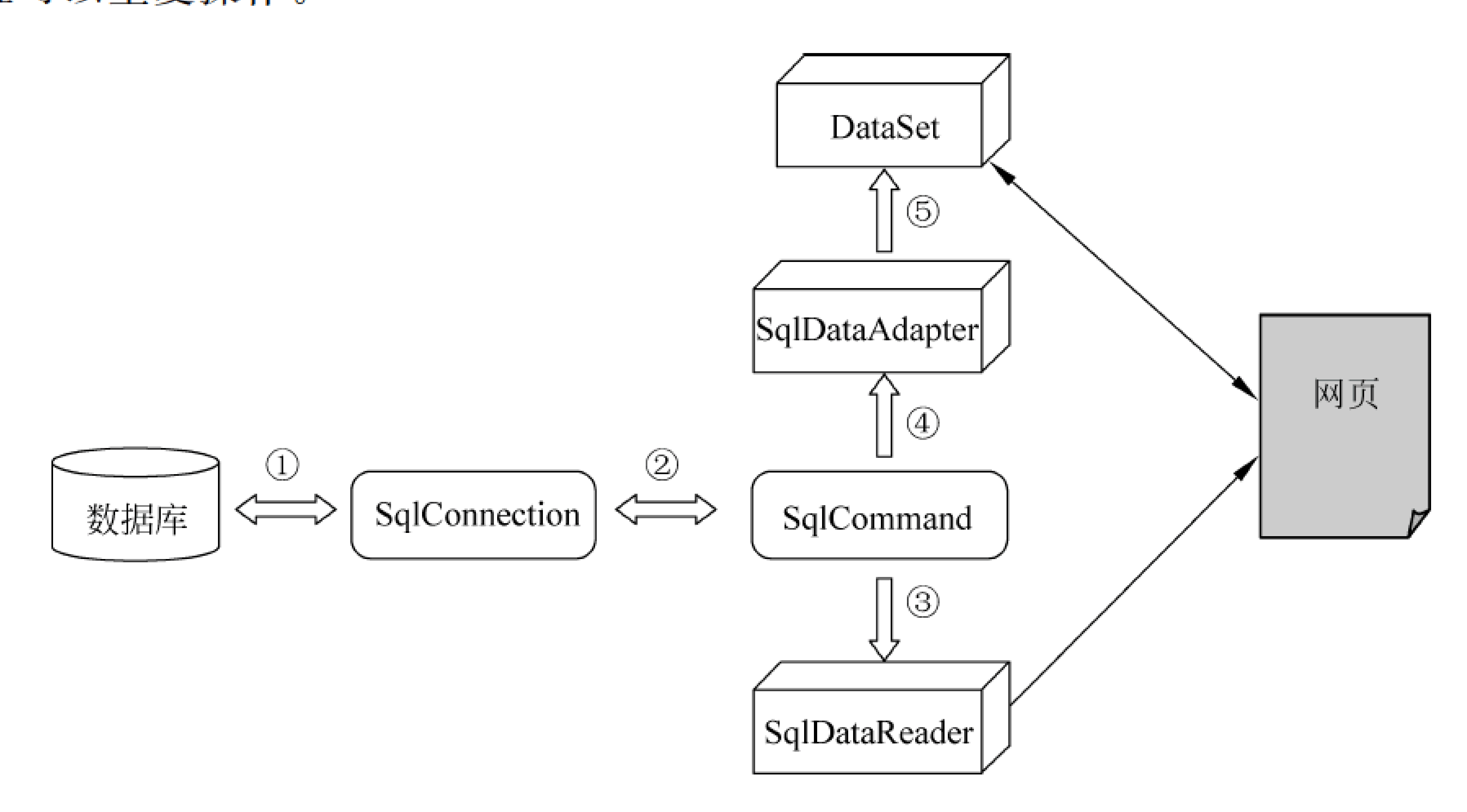
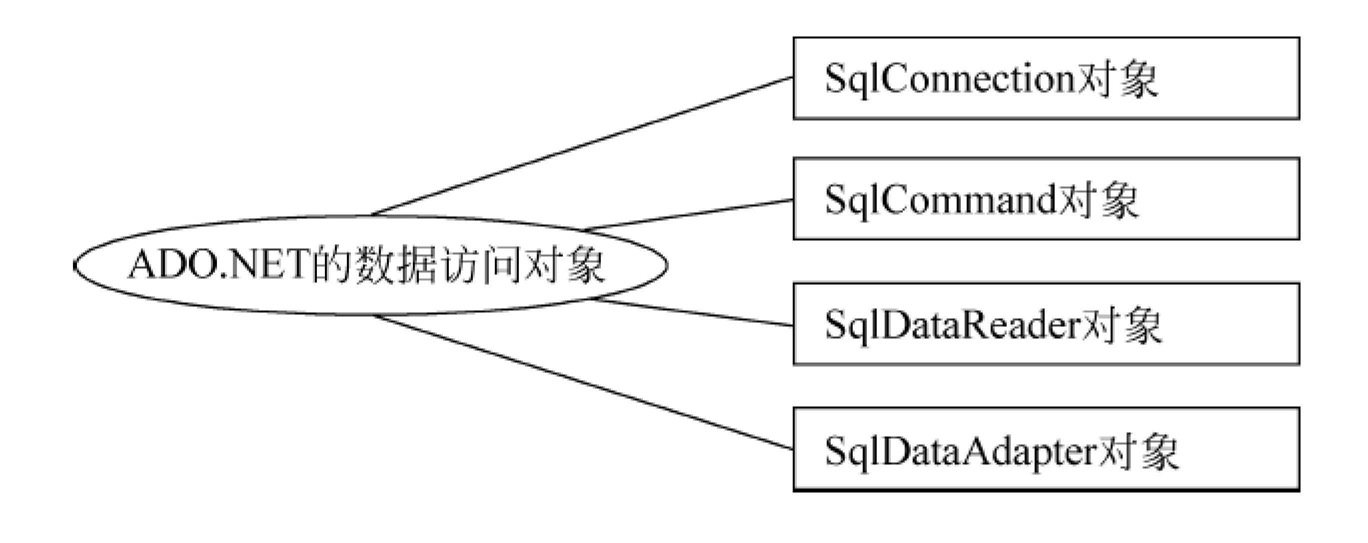


图 10.8　ADO .NET 访问 SQL Server 数据库的一般流程

10.3　ADO .NET 的数据访问对象

知识梳理



10.3.1　SqlConnection 对象

ADO .NET 的数据访问对象有 SqlConnection、SqlCommand、SqlDataReader 和 SqlDataAdapter 等。这里先介绍 SqlConnection 对象。

在数据访问中首先必须是建立数据库的物理连接。SQL .NET Data Provider 使用 SqlConnection 类对象标识与一个数据库的物理连接。

1. SqlConnection 对象的属性和方法

SqlConnection 对象的常用属性如表 10.4 所示。State 枚举成员值如表 10.5 所示。其常用的方法如表 10.6 所示。

表 10.4　SqlConnection 对象的常用属性及其说明

属　性	说　明
ConnectionString	获取或设置用于打开数据库的字符串
ConnectionTimeout	获取在尝试建立连接时终止尝试并生成错误之前所等待的时间
Database	获取当前数据库或连接打开后要使用的数据库的名称
DataSource	获取数据源的服务器名或文件名
Provider	获取在连接字符串的"Provider="子句中指定的 SQL 提供程序的名称
State	获取所建连接的当前状态。其取值及其说明如表 10.5 所示，这些枚举成员值位于 System.Data 命名空间

表 10.5　State 枚举成员值

枚举成员	说　明
Broken	与数据源的连接中断。只有在连接打开之后才可能发生这种情况。可以关闭处于这种状态的连接，然后重新打开
Closed	连接处于关闭状态
Connecting	连接对象正在与数据源连接
Executing	连接对象正在执行命令
Fetching	连接对象正在检索数据
Open	连接处于打开状态

表 10.6　SqlConnection 对象的常用方法及其说明

方　法	说　明
Open	使用 ConnectionString 所指定的属性设置打开数据库连接
Close	关闭与数据库的连接。这是关闭任何打开连接的首选方法
CreateCommand	创建并返回一个与 SqlConnection 关联的 SqlCommand 对象
ChangeDatabase	为打开的 SqllConnection 更改当前数据库

2. 设置 SqlConnection 对象的连接字符串 ConnectionString 属性

创建连接需要使用 SqlConnection 对象，最重要的是设置 SqlConnection 对象的 ConnectionString 连接字符串属性。建立连接字符串的方式是，创建一个 SqlConnection 对象，将 ConnectionString 属性设置为如下值(SQL Server 采用 Windows 身份验证模式)：

```
Data Source = LCB - PC;Initial Catalog = school;Integrated Security = True
```

如果 SQL Server 采用 SQL Server 身份验证模式，登录账户为 sa，密码为 12345，则 ConnectionString 属性设置为如下值：

```
"Data Source = localhost;Initial Catalog = Stud;
    Integrated Security = False;User Id = sa;Password = 12345"
```

ConnectionString 属性中的常用的关键字值有效名称如下：

- Data Source(或 Server 或 Address)：要连接的 SQL Server 实例名称或服务器名称。
- Initial Catalog(或 Database)：连接的数据库的名称。
- Integrated Security：当为 False(默认值)时，将在连接中指定用户 ID 和密码。当为 True 时，将使用当前的 Windows 账户凭据进行身份验证。可识别的值为 True、False、yes、no 以及与 True 等效值。
- User ID：SQL Server 登录账户。
- Password(或 Pwd)：SQL Server 账户登录的密码。

在指定连接字符串后，最后用 Open 方法打开连接。

【练一练】 在 D 盘的电子商务目录中建立一个 CH10 的子目录，将其作为网站目录，设计一个 webform1 网页，其功能是说明创建连接的方法。

其设计步骤如下：



图 10.9　webform1 网页设计界面

① 打开 CH10 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH10”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform1. aspx，其他保持默认项，单击“添加”按钮。

② 其设计界面如图 10.9 所示，其中包含一个 Button1 控件和一个标签 Label1。在该网页上设计如下事件过程：

```
using System.Data.SqlClient;
protected void Button1_Click(object sender, EventArgs e)
{   string mystr;
    SqlConnection myconn  =  new SqlConnection();
    mystr  =  "Data Source = LCB - PC;Initial Catalog = school;Integrated Security = True";
    myconn.ConnectionString  =  mystr;
    myconn.Open();
    if (myconn.State  ==  System.Data.ConnectionState.Open)
        Label1.Text  =  "成功连接到 SQL Server 数据库";
    else
        Label1.Text  =  "不能连接到 SQL Server 数据库";
    myconn.Close();
}
```

注意：在本网页和后面所有示例中的后台代码引用部分添加语句“using System. Data. SqlClient;”。

③ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，单击“连接”按钮，其执行结果如图 10.10 所示，表示成功连接到 school 数据库。



图 10.10　webform1 网页运行界面

3. 将连接字符串存放在 web. config 文件中

可以在 web. config 文件中保存用于连接数据库的连

接字符串，再通过对 web. config 文件加密，从而达到保护连接字符串的目的。例如，在<configuration>节中插入以下代码：

```
<connectionStrings>
    <add name = "myconnstring"
       connectionString = "Data Source = LCB - PC;Initial Catalog = school;
           Integrated Security = True"
    providerName = "System.Data.SqlClient" />
</connectionStrings>
```

这样，以下代码自动获取 web. config 文件中的连接字符串 myconnstring 并打开连接，后面的示例均采用这种方法：

```
string mystr = System.Configuration.ConfigurationManager.
                 ConnectionStrings["myconnstring"].ToString();
SqlConnection myconn = new SqlConnection();
myconn.ConnectionString = mystr;
myconn.Open();
```

10.3.2　SqlCommand 对象

建立数据连接之后，就可以执行数据访问操作和数据操纵操作了。一般对数据库的操作被概括为 CRUD——Create、Read、Update 和 Delete。在 ADO .NET 中定义 SqlCommand 类对象去执行这些操作。

1. SqlCommand 对象的属性和方法

SqlCommand 对象有自己的属性，其属性包含对数据库执行命令所需要的全部信息，通常包括以下内容：

- 一个连接：命令引用一个连接，使用它与数据库通信。
- 命令的名称或文本：包含某 SQL 语句的实际文本或要执行存储过程的名称。
- 命令类型：指明命令的类型，如命令是存储过程还是普通的 SQL 文本。
- 参数：命令可能要求随命令传递参数，命令还可能返回值或通过输出参数的形式返回值。每个命令都有一个参数集合，可以分别设置或读取这些参数以传递或接受值。

SqlCommand 对象的常用属性如表 10.7 所示，CommandType 枚举成员值及说明如表 10.8 所示，其常用方法如表 10.9 所示。

表 10.7　SqlCommand 对象的常用属性及其说明

属　性	说　明
CommandText	获取或设置要对数据源执行的 SQL 语句或存储过程
CommandTimeout	获取或设置在终止执行命令的尝试并生成错误之前的等待时间
CommandType	获取或设置一个值，该值指示如何解释 CommandText 属性。其取值如表 10.8 所示
Connection	获取或设置 SqlCommand 的此实例使用的 SqlConnection
Parameters	参数集合(SqlParameterCollection)

表 10.8　CommandType 枚举成员值

枚举成员	说　明
StoredProcedure	CommandType 属性应设置为存储过程的名称。如果存储过程名称包含任何特殊字符，则可能会要求用户使用转义符语法。当调用 Execute 方法之一时，该命令将执行此存储过程
TableDirect	CommandType 属性应设置为要访问的表的名称。当调用 Execute 方法之一时，将返回命名表的所有行和列
Text	SQL 文本命令（默认）。不支持在向通过 Text 的 CommandType 调用的 SQL 语句或存储过程传递参数时使用问号（?）占位符。在这种情况下，必须使用命名的参数

表 10.9　SqlCommand 对象的常用方法及其说明

方法	说　明
CreateParameter	创建 SqlParameter 对象的新实例
ExecuteNonQuery	针对 Connection 执行 SQL 语句并返回受影响的行数
ExecuteReader	将 CommandText 发送到 Connection 并生成一个 SqlDataReader
ExecuteScalar	执行查询，并返回查询所返回的结果集中第一行的第一列，而忽略其他列或行

2. 创建 SqlCommand 对象

SqlCommand 对象的主要构造函数如下：

```
SqlCommand();
SqlCommand(cmdText);
SqlCommand(cmdText,connection);
```

其中，cmdText 参数指定查询的文本。connection 参数是一个 SqlConnection，表示到 SQL Server 数据库的连接。例如，以下语句创建一个 SqlCommand 对象 mycmd：

```
SqlConnection myconn = new SqlConnection();
string mystr = System.Configuration.ConfigurationManager.
               ConnectionStrings["myconnstring"].ToString();
myconn.ConnectionString = mystr;
myconn.Open();
SqlCommand mycmd = new SqlCommand("SELECT * FROM student",myconn);
```

3. 通过 SqlCommand 对象返回单个值

在 SqlCommand 的方法中，ExecuteScalar 方法执行返回单个值的 SQL 命令。例如，如果想获取 student 表中学生的总人数，则可以使用这个方法执行 SQL 查询命令“SELECT Count(*) FROM student”。

【练一练】 在本章 CH10 网站中添加一个 webform2 网页，其功能是求 student 表中指定班的学生人数。



图 10.11　webform2 网页的设计界面

其设计步骤如下：

① 打开 CH10 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH10”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform2. aspx，其他保持默认项，单击“添加”按钮。

② 其设计界面如图 10.11 所示，其中包含一个 HTML

文字、一个文本框 TextBox1、一个 Button1 控件和一个标签 Label1。在该网页上设计如下事件过程：

```
protected void Button1_Click(object sender, EventArgs e)
{   string mystr, mysql;
    int rs;
    SqlConnection myconn = new SqlConnection();
    SqlCommand mycmd = new SqlCommand();
    mystr = System.Configuration.ConfigurationManager.
            ConnectionStrings["myconnstring"].ToString();
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT COUNT( * ) FROM student WHERE 班号 = '" + TextBox1.Text.Trim() + "'";
    mycmd.CommandText = mysql;
    mycmd.Connection = myconn;
    rs = int.Parse(mycmd.ExecuteScalar().ToString());
    if (rs == 0)
        Label1.Text = "你输入的班号不存在";
    else
        Label1.Text = TextBox1.Text.Trim() + "班人数为" + rs.ToString();
     myconn.Close();
}
```

上述代码先建立连接 myconn，然后创建 SqlCommand 对象 mycmd，并设置要执行的 SQL 语句。再使用 mycmd 对象的 ExecuteScalar 方法返回聚合函数的执行结果，即指定班的学生人数，它是单个值。最后关闭连接。

③ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，输入班号为 15002，单击"求人数"按钮，其执行结果如图 10.12 所示。



图 10.12　webform2 网页的执行界面

4. 通过 SqlCommand 对象执行更新操作

在 SqlCommand 的方法中，ExecuteNonQuery 方法执行不返回结果的 SQL 命令。该方法主要用来更新数据，通常使用它来执行 UPDATE、INSERT 和 DELETE 语句。该方法不返回行，对于 UPDATE、INSERT 和 DELETE 语句，返回值为该命令所影响的行数，对于所有其他类型的语句，返回值为－1。

【练一练】　在本章 CH10 网站中添加一个 webform3 网页，其功能是说明通过 SqlCommand 对象执行更新操作的方法。



图 10.13　webform3 网页的设计界面

其设计步骤如下：

① 打开 CH10 网站，选择"网站|添加新项"菜单命令，出现"添加新项-CH10"对话框，在中间列表中选择"Web 窗体"，将文件名称改为 webform3.aspx，其他保持默认项，单击"添加"按钮。

② 其设计界面如图 10.13 所示，其中包含一个 Button1 控件和一个标签 Label1。在该网页上设计如下事件过程：

```
protected void Button1_Click(object sender, EventArgs e)
{   string mystr, mysql;
    SqlConnection myconn = new SqlConnection();
```

```
        SqlCommand mycmd = new SqlCommand();
        mystr = System.Configuration.ConfigurationManager.
            ConnectionStrings["myconnstring"].ToString();
        myconn.ConnectionString = mystr;
        myconn.Open();
        mysql = "INSERT INTO student VALUES(88,'许涛','男','汉族','15005')";
        mycmd.CommandText = mysql;
        mycmd.Connection = myconn;
        mycmd.ExecuteNonQuery();
        Label1.Text = "(1)插入学号为 88 的学生记录<br>";
        mysql = "DELETE student WHERE 学号 = 88";
        mycmd.CommandText = mysql;
        mycmd.Connection = myconn;
        mycmd.ExecuteNonQuery();
        Label1.Text += "(2)删除学号为 88 的学生记录";
        myconn.Close();
    }
```

上述代码先建立连接 myconn，然后创建 SqlCommand 对象 mycmd，并设置要执行的 INSERT 语句，再使用 mycmd 对象的 ExecuteNonQuery 方法执行该 INSERT 语句，在 student 表中插入一个学号为 88 的学生记录。接着给 mycmd 对象设置新的 DELETE 语句，同样使用 mycmd 对象的 ExecuteNonQuery 方法执行该 DELETE 语句，删除刚插入的记录。最后关闭连接。



图 10.14　webform3 网页的执行界面

③ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，单击“插入和删除”命令按钮，其执行结果如图 10.14 所示。

进入 SQL Server，打开 school 数据库的 student 表，看到所有记录没有发生改变。

5. 在 SqlCommand 对象的命令中指定参数

SQL .NET Data Provider 支持执行命令中包含参数的情况，也就是说，可以使用包含参数的数据命令或存储过程执行数据筛选操作和数据更新等操作，其主要流程如下：

① 创建 Connection 对象，并设置相应的属性值。

② 打开 Connection 对象。

③ 创建 Command 对象并设置相应的属性值，其中 SQL 语句参数使用参数名称。

④ 创建参数对象，将建立好的参数对象添加到 Command 对象的 Parameters 集合中。

⑤ 为参数对象赋值。

⑥ 执行数据命令。

⑦ 关闭相关对象。

例如，下面的更新语句：

```
UPDATE customer SET cName = @Name WHERE cID = @ID
```

其中 customer 是一个顾客表，有 cID(顾客号)和 cName(顾客名)两个列。该命令是将指定 cID 的顾客记录的 cName 替换成指定的值。其中@ID 和@Name 均为参数，在执行该语句之前需要为参数赋值。

如何为参数赋值呢？SqlCommand 对象的 Parameters 属性能够取得与 Sqlcommand 相关

联的参数集合(也就是 SqlParameterCollection),从而通过调用其 Add 方法即可将 SQL 语句中的参数添加到参数集合中,每个参数都是一个 SqlParameter 类对象,其常用属性及说明如表 10.10 所示。

表 10.10　SqlParameter 的常用属性及其说明

属性	说　明
ParameterName	用于指定参数的名称
SqlDbType	用于指定参数的数据类型,如整型、字符型等
Value	设置输入参数的值
Size	设置数据的最大长度(以字节为单位)
Scale	设置小数位数
Direction	指定参数的方向,可以是下列值之一。ParameterDirection. Input:指明为输入参数。ParameterDirection. Output:指明为输出参数。ParameterDirection. InputOutput:指明为输入参数或者输出参数。ParameterDirection. ReturnValue:指明为返回值类型

例如,假设 mycmd 数据命令对象包含前面带参数的命令,可以使用以下命令向 Parameters 参数集合中添加参数值:

```
mycmd.Parameters.Add("@Name",SqlDbType.VarChar,10).Value = Name1;
mycmd.Parameters.Add("@ID", SqlDbType.VarChar,5).Value = ID1;
```

上面 Add 方法中的第 1 个参数为参数名,第 2 个参数为参数的数据类型,第 3 个参数为参数值的最大长度,并分别将参数值设置为 Name1 和 ID1 变量。上述语句也可以等价地改为:

```
SqlParameter myparm1 = new SqlParameter();
myparm1.ParameterName = "@Name";
myparm1.SqlDbType = SqlDbType.VarChar;
myparm1.Size = 10;
myparm1.Value = Name1;                          //设置参数值
mycmd.Parameters.Add(myparm1);
SqlParameter myparm2 = new SqlParameter();
myparm2.ParameterName = "@ID";
myparm2.SqlDbType = SqlDbType.VarChar;
myparm2.Size = 5;
myparm2.Value = ID1;                            //设置参数值
mycmd.Parameters.Add(myparm2);
```

【练一练】 在本章 CH10 网站中添加一个 webform4 网页,其功能是说明 SqlCommand 对象的命令中参数的使用方法。

其设计步骤如下:

① 打开 CH10 网站,选择“网站|添加新项”菜单命令,出现“添加新项-CH10”对话框,在中间列表中选择“Web 窗体”,将文件名称改为 webform4.aspx,其他保持默认项,单击“添加”按钮。

② 其设计界面如图 10.15 所示,其中包含两个 HTML 文字、两个文本框(TextBox1 和 TextBox2)、一个 Button1 控件和一个标签 Label1。在该网页

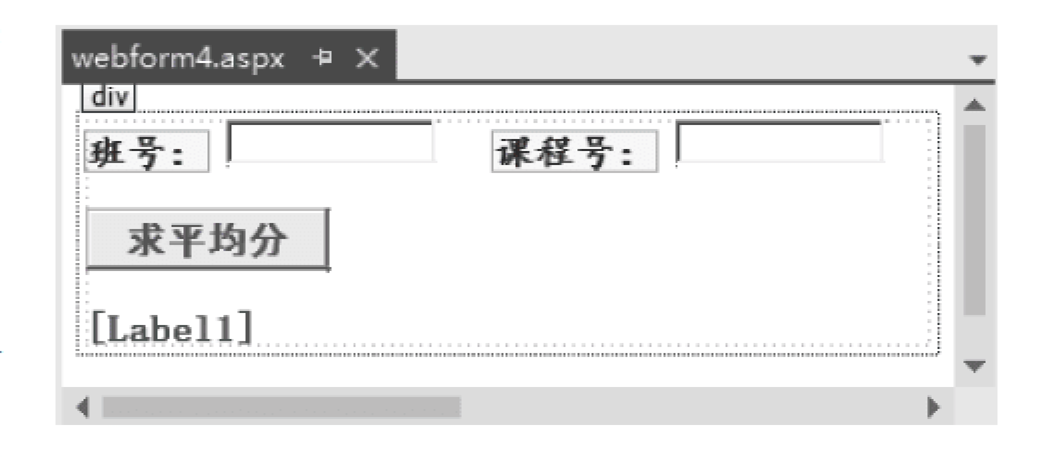


图 10.15　webform4 网页的设计界面

上设计如下事件过程：

```
protected void Button1_Click(object sender, EventArgs e)
{   string mystr, mysql;
    SqlConnection myconn = new SqlConnection();
    SqlCommand mycmd = new SqlCommand();
    mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myconnstring"].ToString();
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT AVG(score.分数) ";
    mysql += "FROM student,score ";
    mysql += "WHERE student.学号 = score.学号 ";
    mysql += "AND student.班号 = @bh AND score.课程号 = @kch";
    mycmd.CommandText = mysql;
    mycmd.Connection = myconn;
    mycmd.Parameters.Add("@bh", System.Data.SqlDbType.VarChar, 10).Value
         = TextBox1.Text;
    mycmd.Parameters.Add("@kch", System.Data.SqlDbType.VarChar, 10).Value
         = TextBox2.Text;
    try
    {   Label1.Text = TextBox1.Text + "班" + TextBox2.Text + "课程的平均分为";
        float avg = float.Parse(mycmd.ExecuteScalar().ToString());
        Label1.Text += Math.Round(avg, 2);
    }
    catch (Exception ex)
    {
        Label1.Text = "提示：输入的班号或课程号不存在";
    }
    myconn.Close();
}
```

上述代码先建立连接，然后创建一个 SqlCommand 对象 mycmd，将其 CommandText 属性设置为带两个参数的 SELECT 语句，在 mycmd.Parameters 参数集合属性中添加两个参数对象，最后通过调用 ExecuteScalar 方法执行该 SELECT 语句，并返回指定班号和课程号的平均分。try-catch 语句用于检测用户输入不存在的班号或课程号的异常情况。

③ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，输入正确的班号和课程号，单击“求平均分”按钮，其执行结果如图 10.16 所示。若输入不存在的班号或课程号，单击“求平均分”按钮，其执行结果如图 10.17 所示。



图 10.16　webform4 网页的执行界面一



图 10.17　webform4 网页的执行界面二

10.3.3　SqlDataReader 对象

当执行 SQL 语句返回结果集的命令时，需要一个方法从结果集中提取数据。处理结果集的方法有两个：第一，使用 SqlDataReader 对象(数据阅读器)；第二，同时使用 SqlDataAdapter 对象(数据适配器)和 DataSet 对象。

不过，使用 DataReader 对象可以从数据库中得到只读的、只能向前的数据流。使用 SqlDataReader 对象还可以提高应用程序的性能，减少系统开销，因为同一时间只有一条行记录在内存中。

1. SqlDataReader 类的属性和方法

SqlDataReader 类的常用属性如表 10.11 所示。其常用方法如表 10.12 所示。

表 10.11　SqlDataReader 类的常用属性及其说明

属性	说　明
FieldCount	获取当前行中的列数
IsClosed	获取一个布尔值，指出 SqlDataReader 对象是否关闭
RecordsAffected	获取执行 SQL 语句时修改的行数

表 10.12　SqlDataReader 类的常用方法及其说明

方法	说　明
Read	将 SqlDataReader 对象前进到下一行并读取，返回布尔值指示是否有多行
Close	关闭 SqlDataReader 对象
IsDBNull	返回布尔值，表示列是否包含 NULL 值
NextResult	将 SqlDataReader 对象移到下一个结果集，返回布尔值指示该结果集是否有多行
GetBoolean	返回指定列的值，类型为布尔值
GetString	返回指定列的值，类型为字符串
GetByte	返回指定列的值，类型为字节
GetInt32	返回指定列的值，类型为整型值
GetDouble	返回指定列的值，类型为双精度值
GetDataTime	返回指定列的值，类型为日期时间值
GetOrdinal	返回指定列的序号或数字位置(首列序号为 0)
GetBoolean	返回指定列的值，类型为对象

2. 创建 SqlDataReader 对象

在 ADO .NET 中从来不会显式地使用 SqlDataReader 对象的构造函数创建的 SqlDataReader 对象。事实上，SqlDataReader 类没有提供公有的构造函数。人们通常调用 Command 类的 ExecuteReader 方法，这个方法将返回一个 SqlDataReader 对象。例如，以下代码创建一个 SqlDataReader 对象 myreader：

```
SqlCommand cmd = new SqlCommand(CommandText, ConnectionObject);
SqlDataReader myreader = cmd.ExecuteReader();
```

注意： SqlDataReader 对象不能使用 new 来创建。

SqlDataReader对象最常见的用法就是检索SQL查询或存储过程执行后返回的记录集。另外,SqlDataReader是一个连接、只向前和只读的记录集。也就是说,当使用数据阅读器时,必须保持连接处于打开状态。此外,可以从头到尾浏览记录集,而且也只能以这样的次序浏览。这就意味着,不能在某条记录处停下来向之前的记录移动。记录是只读的,因此数据阅读器类不提供任何修改数据库记录的方法。

注意: SqlDataReader对象使用底层的连接,连接是它专有的。当SqlDataReader对象打开时,不能使用对应的连接对象执行其他任何任务,如执行另外的命令等。当不再需要SqlDataReader对象的记录时,应该立刻关闭它。

3. 遍历SqlDataReader对象的记录

当ExecuteReader方法返回SqlDataReader对象时,当前光标的位置是第一条记录的前面。必须调用SqlDataReader对象的Read方法把光标移动到第一条记录,然后,第一条记录将变成当前记录。如果SqlDataReader对象中包含的记录不止一条,Read方法就返回一个Boolean值True。想要移动到下一条记录,需要再次调用Read方法。重复上述过程,直到最后一条记录,此时Read方法将返回False。经常使用While循环来遍历记录:

```
while (myreader.Read())
{
    //读取数据
}
```

只要Read方法返回的值为True,就可以访问当前记录中包含的字段。

4. 访问字段中的值

使用以下语句获取一个SqlDataReader对象:

```
SqlDataReader myreader = mycmd.ExecuteReader();
```

然后ADO.NET提供了两种方法来访问记录中的字段。第一种是Item属性,此属性返回由字段索引或字段名指定的字段值;第二种方法是Get方法,此方法返回由字段索引指定的字段的值。

(1) Item属性

每一个SqlDataReader对象都定义了一个Item属性,此属性返回一个代码字段属性的对象。Item属性是SqlDataReader对象的索引。需要注意的是,Item属性总是基于0开始编号的:

```
myreader[FieldName]
myreader[FieldIndex]
```

可以把包含字段名的的字符串传递给Item属性,也可以把指定字段索引的32位整数传递给Item属性。例如,如果命令是SELECT查询:

```
SELECT ID,cName FROM customer
```

使用下面任意一种方法,都可以得到两个被返回字段的值:

```
myreader["ID"]
myreader["cName"]
```

或

```
myreader[0]
myreader[1]
```

（2）Get 方法

每一个 SqlDataReader 对象都定义了一组 Get 方法，那些方法将返回适当类型的值。例如，GetInt32 方法把返回的字段值作为 32 位整数，每一个 Get 方法都将接受字段的索引。例如，在上面的例子中，使用以下的代码可以检索 ID 字段和 cName 字段的值：

```
myreader.GetInt32(0)
myreader.GetString(1)
```

【练一练】 在本章 CH10 网站中添加一个 webform5 网页，其功能是在列表框中显示指定课程号的学生学号、姓名和分数，并按分数递减排列，以此说明 SqlDataReader 对象的使用方法。



图 10.18　webform5 网页的设计界面图

其设计步骤如下：

① 打开 CH10 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH10”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform5.aspx，其他保持默认项，单击“添加”按钮。

② 其设计界面如图 10.18 所示，其中包含一个 HTML 文字、一个文本框 TextBox1、一个 Button1 控件和一个列表框 ListBox1。在该网页上设计如下事件过程：

```
protected void Button1_Click(object sender, EventArgs e)
{   string mystr, mysql;
    SqlConnection myconn = new SqlConnection();
    SqlCommand mycmd = new SqlCommand();
    mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myconnstring"].ToString();
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT student.学号,student.姓名,score.分数 ";
    mysql += "FROM student,score ";
    mysql += "WHERE student.学号 = score.学号 AND score.课程号 = '"
          + TextBox1.Text + "'";
    mysql += "ORDER BY score.分数 DESC";
    mycmd.CommandText = mysql;
    mycmd.Connection = myconn;
    SqlDataReader myreader = mycmd.ExecuteReader();
    ListBox1.Items.Add("学号 姓名 分数");
    ListBox1.Items.Add(" ================= ");
    while (myreader.Read())                          //循环读取信息
        ListBox1.Items.Add(String.Format("{0} {1} {2}",
            myreader["学号"].ToString(), myreader[1].ToString(),
            myreader.GetDouble(2)));
    myconn.Close();
    myreader.Close();
}
```

上述代码通过执行 SqlCommand 对象的 ExecuteReader()方法，将结果集放置在一个 SqlDataReader 对象 myreader 中，然后从 myreader 对象读取一行一行的记录显示在列表框 ListBox1 中。

③ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，输入正确的课程号 201，单击“成绩列表”按钮，其执行结果如图 10.19 所示。



图 10.19 webform5 网页的执行界面

10.3.4 SqlDataAdapter 对象

SqlDataAdapter 对象(数据适配器)可以执行 SQL 命令以及调用存储过程、传递参数，最重要的是取得数据结果集，在数据库和 DataSet 对象之间来回传输数据。

1. SqlDataAdapter 类的属性和方法

SqlDataAdapter 类的常用属性如表 10.13 所示。其常用方法如表 10.14 所示。

表 10.13 SqlDataAdapter 类的常用属性及其说明

属性	说 明
SelectCommand	获取或设置 SQL 语句用于选择数据源中的记录
InsertCommand	获取或设置 SQL 语句用于将新记录插入到数据源中
UpdateCommand	获取或设置 SQL 语句用于更新数据源中的记录
DeleteCommand	获取或设置 SQL 语句用于从数据集中删除记录
AcceptChangesDuringFill	获取或设置一个值，该值指示在任何 Fill 操作过程中时，是否接受对行所做的修改
AcceptChangesDuringUpdate	获取或设置在 Update 期间是否调用 AcceptChanges
FillLoadOption	获取或设置 LoadOption，后者确定适配器如何从 SqlDataReader 中填充 DataTable
MissingMappingAction	确定传入数据没有匹配的表或列时需要执行的操作
MissingSchemaAction	确定现有 DataSet 架构与传入数据不匹配时需要执行的操作
TableMappings	获取一个集合，它提供源表和 DataTable 之间的主映射

表 10.14 SqlDataAdapter 类的常用方法及其说明

方法	说 明
Fill	用来自动执行 SqlDataAdapter 对象的 SelectCommand 属性中相对应的 SQL 语句，以检索数据库中的数据，然后更新数据集中的 DataTable 对象，如果 DataTable 对象不存在，则创建它
FillSchema	将 DataTable 添加到 DataSet 中，并配置架构以匹配数据源中的架构
GetFillParameters	获取当执行 SELECT 语句时由用户设置的参数
Update	用来自动执行 UpdateCommand、InsertCommand 或 DeleteCommand 属性相对应的 SQL 语句，以使数据集中的数据来更新数据库

实际上，使用 SqlDataAdapter 对象的主要目的是取得 DataSet 对象。另外，它还有一个功能，就是数据写回更新的自动化。DataSet 对象为离线存取，因此数据的添加、删除、修改都在 DataSet 中进行，当需要数据批次写回数据库时，SqlDataAdapter 对象提供了一个 Update 方

法，它会自动将 DataSet 中不同的内容取出，然后自动判断添加的数据并使用 InsertCommand 所指定的 INSERT 语句，修改的记录使用 UpdateCommand 所指定的 UPDATE 语句，以及删除的记录使用 DeleteCommand 指定的 DELETE 语句来更新数据库的内容。

在写回数据来源时，DataTable 与实际数据的数据表及列的对应，则可以通过 TableMappings 定义对应关系。

2. 创建 SqlDataAdapter 对象

SqlDataAdapter 类有以下构造函数：

```
SqlDataAdapter();
SqlDataAdapter(selectCommandText);
SqlDataAdapter(selectCommandText,selectConnection);
SqlDataAdapter((selectCommandText,selectConnectionString);
```

其中，selectCommandText 是一个字符串，包含 SQL SELECT 语句或存储过程。selectConnection 是当前连接的 SqlConnection 对象。selectConnectionString 是连接字符串。

采用上述第 3 个构造函数创建 SqlDataAdapter 对象的过程是：先建立 SqlConnection 连接对象，接着建立 SqlDataAdapter 对象，建立该对象的同时可以传递两个参数：命令字符串(mysql)和连接对象(myconn)。

例如：

```
string mystr,mysql;
SqlConnection myconn = new SqlConnection();
mystr = System.Configuration.ConfigurationManager.
    ConnectionStrings["myconnstring"].ToString();
myconn.ConnectionString = mystr;
myconn.Open();
mysql = "SELECT * FROM student";
SqlDataAdapter myadapter = new SqlDataAdapter(mysql,myconn);
myconn.Close();
```

以上代码仅创建了 SqlDataAdapter 对象，没有使用它，在后面介绍 DataSet 对象时大量使用 SqlDataAdapter 对象。

3. 使用 Fill 方法

Fill 方法用于向 DataSet 对象填充从数据源中读取的数据。调用 Fill 方法的语法格式有多种，常见的格式如下：

```
SqlDataAdapter 对象名.Fill(DataSet 对象名,"数据表名");
```

其中，第一个参数是数据集对象名，表示要填充的数据集对象；第二个参数是一个字符串，表示在本地缓冲区中建立的临时表的名称。例如，以下语句用 customer 表数据填充数据集 mydataset1：

```
SqlDataAdapter1.Fill(mydataset1,"customer");
```

使用 Fill 方法要注意以下几点：

- 如果调用 Fill()之前连接已关闭，则先将其打开以检索数据，数据检索完成后再将连接关闭。如果调用 Fill()之前连接已打开，连接仍然会保持打开状态。
- 如果数据适配器在填充 DataTable 时遇到重复列，它们将以 columnname1、columnname2、

columnname3…形式命名后面的列。

- 如果传入的数据包含未命名的列，它们将以 column1、column2 形式命名并存入 DataTable。
- 向 DataSet 添加多个结果集时，每个结果集都放在一个单独的表中。
- 可以在同一个 DataTable 中多次使用 Fill()方法。如果存在主键，则传入的行会与已有的匹配行合并；如果不存在主键，则传入的行会追加到 DataTable 中。

4. 使用 Update 方法

Update 方法用于将数据集 DataSet 对象中的数据按 InsertCommand 属性、DeleteCommand 属性和 UpdateCommand 属性所指定的要求更新数据源，即调用 3 个属性中所定义的 SQL 语句来更新数据源。

Update 方法常见的调用格式如下：

```
SqlDataAdapter 对象名.Update(DataSet 对象名,[数据表名]);
```

其中，第一个参数是数据集对象名，表示要将哪个数据集对象中的数据更新到数据源中；第二个参数是一个字符串，表示临时表的名称。

由于 SqlDataAdapter 对象介于 DataSet 对象和数据源之间，Update 方法只能将 DataSet 中的修改回存到数据源中，有关修改 DataSet 对象中数据的方法将在后面介绍。当用户修改 DataSet 对象中的数据时，如何产生 SqlDataAdapter 对象的 InsertCommand、DeleteCommand 和 UpdateCommand 属性呢？

系统提供了 SqlCommandBuilder 类，它根据用户对 DataSet 对象数据的操作自动生成相应的 InsertCommand、DeleteCommand 和 UpdateCommand 属性值。该类的构造函数如下：

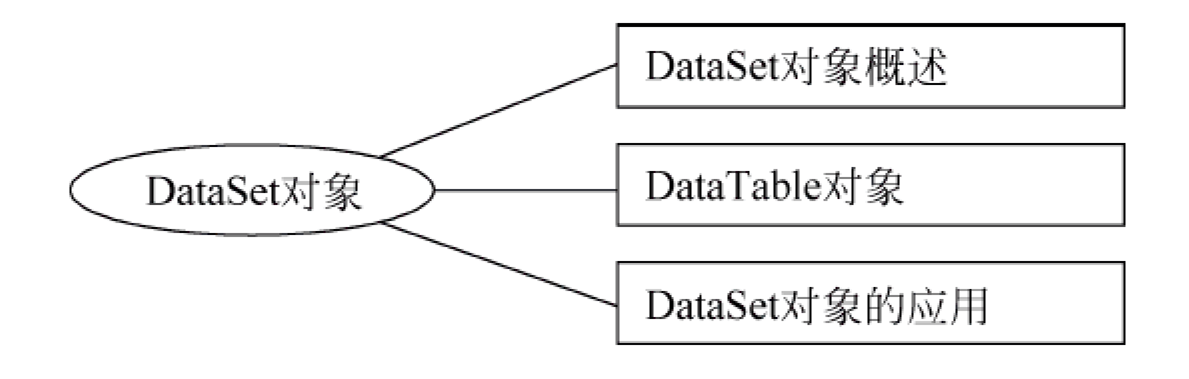
```
SqlCommandBuilder(adapter);
```

其中，adapter 是一个 SqlDataAdapter 对象的名称。例如，以下语句创建一个 SqlCommandBuilder 对象 mycmdbuilder，用于产生 myadp 对象的 InsertCommand、DeleteCommand 和 UpdateCommand 属性值，然后调用 Update 方法执行这些修改命令以更新数据源：

```
SqlCommandBuilder mycmdbuilder = new SqlCommandBuilder(myadp);
myadp.Update(myds, "student");
```

10.4　DataSet 对象

知识梳理



10.4.1 DataSet 对象概述

DataSet 是 ADO .NET 数据库访问组件的核心，主要是用来支持 ADO .NET 的不连贯连接及数据分布。它的数据驻留内存，可以保证和数据源无关的一致的关系模型，用于多个异种数据源的数据操作。

创建 DataSet 对象有多种方法，既可以使用设计工具，也可以使用程序代码来创建 DataSet 对象。使用程序代码创建 DataSet 对象的语法格式如下：

```
DataSet 对象名 = new DataSet();
DataSet 对象名 = new DataSet(dataSetName);
```

其中，dataSetName 为一个字符串，用于指出 DataSet 的名称。

说明：DataSet 对象的命名空间是 System.Data。在使用 DataSet 对象的网页中要使用 using System.Data 语句引用该命名空间。

DataSet 对象如同内存中的数据库，一个 DataSet 对象包含一个 Tables 属性（表集合）和一个 Relations 属性（表之间关系的集合）等集合属性，其中 Tables 属性如图 10.20 所示，它的每个元素代表 DataSet 对象中的一个表。DataSet 对象的方法较少使用，这里不讨论。

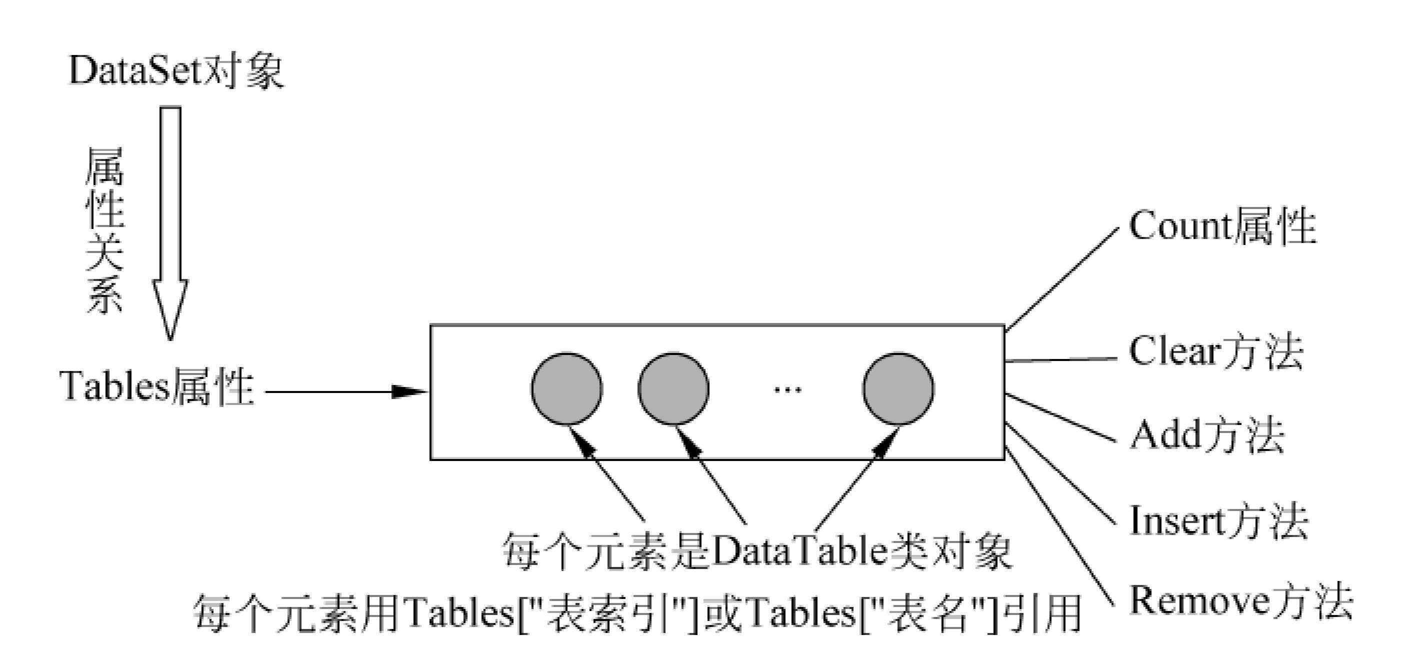


图 10.20 DataSet 对象的 Tables 集合属性

10.4.2 DataTable 对象

DataSet 对象中 Tables 集合属性的每个元素就是一个 DataTable 对象。DataTable 对象的主要属性有 Columns 集合属性（列集合）和 Rows 集合属性（行集合）。

如图 10.21 所示，Columns 集合属性表示 DataSet 对象中一个表的列结构，其中每个元素就是一个 DataColumn 对象，表示表的一个列，每个列有列名和数据类型。例如，如下代码创建一个 DataTable 对象 mydt，并向其中添加 3 个列：

```
DataTable mydt = new DataTable();
DataColumn mycol1 = mydt.Columns.Add("ID", Type.GetType("System.String"));
mydt.Columns.Add("cName", Type.GetType("System.String"));
mydt.Columns.Add("cBook", Type.GetType("System.String"));
```

一个 Rows 集合属性表示 DataSet 对象中一个表的数据（由若干行构成），其中每个元素就是一个 DataRow 对象，表示表的一行。例如，如下代码在 DataSet 对象 myds 中添加一个 customer 的表，该表有 cID 和 cName 两个列，并插入两个记录，如图 10.22 所示。

```
DataSet myds = new DataSet();
DataTable mydt = new DataTable("customer");
myds.Tables.Add(mydt);
mydt.Columns.Add("cID", Type.GetType("System.String"));
mydt.Columns.Add("cName", Type.GetType("System.String"));
DataRow myrow1 = mydt.NewRow();
myrow1["cID"] = "101";
myrow1["cName"] = "章明";
myds.Tables[0].Rows.Add(myrow1);
DataRow myrow2 = mydt.NewRow();
myrow2["cID"] = "120";
myrow2["cName"] = "陈功";
myds.Tables[0].Rows.Add(myrow2);
```

上述代码创建的 DataSet 对象只是存放在内存中，并没有存储在 SQL Server 的数据库中。

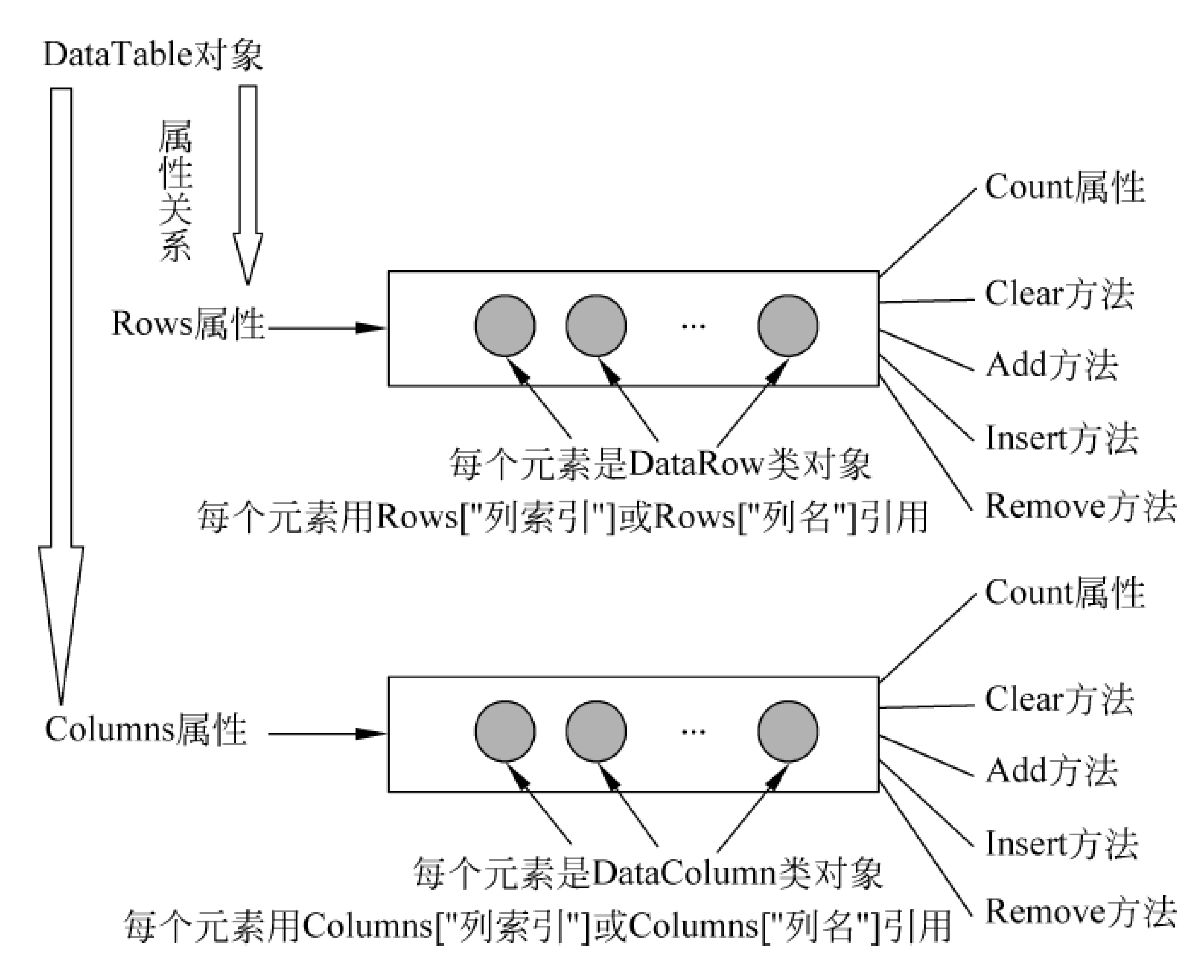


图 10.21　DataTable 对象的 Columns 和 Rows 集合属性

cID	cName
101	章明
120	陈功

图 10.22　采用 DataSet 对象创建的 customer 表

10.4.3　DataSet 对象的应用

DataSet 对象是一个数据容器，可以在内存中存放数据表，而数据来源主要是 SqlDataReader 对象。

【练一练】　在本章 CH10 网站中添加一个 webform6 网页，其功能与 webform5 的相同，说明 DataSet 对象的使用方法。

其设计步骤如下：

① 打开 CH10 网站，选择"网站|添加新项"菜单命令，出现"添加新项-CH10"对话框，在中间列表中选择"Web 窗体"，将文件名称改为 webform6.aspx，其他保持默认项，单击"添加"按钮。

② 其设计界面与 webform5 的完全相同。在该网页上设计如下事件过程：

```
protected void Button1_Click(object sender, EventArgs e)
```

```
{   string mystr, mysql;
    SqlConnection myconn = new SqlConnection();
    mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myconnstring"].ToString();
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT student.学号,student.姓名,score.分数 ";
    mysql += "FROM student,score ";
    mysql += "WHERE student.学号 = score.学号 AND score.课程号 = '"
          + TextBox1.Text + "'";
    mysql += "ORDER BY score.分数 DESC";
    SqlDataAdapter myda = new SqlDataAdapter(mysql,myconn);
    myconn.Close();
    DataSet mydataset = new DataSet();
    myda.Fill(mydataset,"mydata");
    ListBox1.Items.Add("学号 姓名 分数");
    ListBox1.Items.Add(" ================== ");
    foreach (DataRow dr in mydataset.Tables[0].Rows)
        ListBox1.Items.Add(String.Format("{0} {1} {2}",
            dr[0].ToString(), dr[1].ToString(),dr[2].ToString()));
}
```

上述代码创建一个 DataAdapter 对象 myda 和一个 DataSet 对象 mydataset，将 myda 的查询结果通过 Fill 方法填充到 mydataset 中，内存的表名为 mydata(该名称可以是任何合法的表名)。最后使用 foreach 语句将 mydata 的所有行显示在 ListBox1 中。

本网页的执行功能与 webform5 相同。

10.5 练　习　题

1. 单项选择题

(1) 在对 SQL Server 数据库操作时应选用(　　)。

A. SQL Server .NET Framework 数据提供程序
B. OLE DB .NET Framework 数据提供程序
C. ODBC .NET Framework 数据提供程序
D. Oracle .NET Framework 数据提供程序

(2) .NET Framework 数据提供程序包含的对象有(　　)。

A. Connection、Command、DataAdapter 和 DataSet
B. Connection、Command、DataAdapter 和 DataReader
C. Connection、DataSet、DataAdapter 和 DataReader
D. Connection、Command、DataSet 和 DataReader

(3) SqlConnection 对象中用于设置连接 SQL Server 数据库的连接字符串属性是(　　)。

A. DataSource　　B. ServerVersion
C. ConnectionString　　D. State

(4) 用于打开 SqlConnection 对象的方法是(　　)。

A. Select　　B. Find　　C. Open　　D. Close

(5) ADO .NET 中用于建立 Connection 对象和数据集之间桥梁的对象是(　　)。

A. DataTable　　B. Command　　C. DataSet　　D. DataAdapter

(6) SqlCommand 对象中用于设置要对数据源执行 SQL 语句的属性是(　　)。

A. Connection　　B. CommandText　　C. CommandType　　D. Parameters

(7) SqlCommand 对象中将 CommandText 发送到 Connection 并生成一个 SqlDataReader 的方法是(　　)。

A. ExecuteReader　　B. ExecuteScalar

C. ExecuteNonQuery　　D. ToString

(8) 在 ADO .NET 中,对于 Command 对象的 ExecuteNonQuery()方法和 ExecuteReader()方法,下面叙述错误的是(　　)。

A. INSERT、UPDATE、DELETE 等操作的 SQL 语句主要用 ExecuteNonQuery()方法来执行

B. ExecuteNonQuery()方法返回执行 SQL 语句所影响的行数

C. SELECT 操作的 SQL 语句只能由 ExecuteReader()方法来执行

D. ExecuteReader()方法返回一个 DataReder 对象

(9) 以下 SqlComand 对象方法中,可以连接执行 SQL 语句并返回受影响行数的是(　　)。

A. ExecuteReader　　B. ExecuteScala

C. Connection　　D. ExecuteNonQuery

(10) 在 ADO .NET 中,为访问 DataTable 对象从数据源提取的数据行,可使用 DataTable 对象的(　　)属性。

A. Rows　　B. Columns　　C. Constraints　　D. DataSet

(11) 在操作数据库时有查询、更新和删除等操作,在 ADO .NET 中一般使用(　　)对象来完成。

A. Connection　　B. Command　　C. DataAdapter　　D. DataSet

(12) 数据读取器 DataReader 对象用于从数据源中(　　)。

A. 插入数据　　B. 检索数据　　C. 更新数据　　D. 删除数据

(13) 数据适配器 DataAdapter 表示一组(　　)和一个(　　),它们用于填充(　　)对象和更新数据源。

A. 数据命令,数据库连接,Command　　B. 数据命令,数据库连接,DataSet

C. 数据库连接,数据命令,DataSet　　D. 以上都不对

(14) DataAdapter 对象填充 DataSet 对象时使用(　　)方法。

A. Fill　　B. Update　　C. Insert　　D. Delete

(15) DataAdapter 对象的(　　)方法用于更新数据源。

A. Fill　　B. Update　　C. Insert　　D. Delete

(16) 创建 DataAdapter 对象方式错误的是(　　),其中 mysql 是 SQL 语句,myconn 是连接对象。

A. SqlDataAdapter myadapter = new SqlDatadapter();

B. SqlDataAdapter myadapter = new SqlDataDapter();

C. SqlDataAdapter myadapter = new SqlDataAdapter(mysql,myconn);

D. SqlDataAdapter myadapter = new SqlDataAdapter(myconn);

(17) 设计 ADO .NET 应用程序时，在下列情况下，使用 Command 对象直接访问数据源效率最差的是(　　)。

A. 使用 ADO .NET 对 XML 数据文件中的数据进行分析和处理
B. 在 SQL Server 数据库的表中搜索某个字段值
C. 计算 SQL Server 数据库中数据表的行数
D. 在 SQL Server 数据库中创建存储过程

(18) 如果用一个 DataAdapter 对象填充 DataSet 对象，再使用该 DataSet 中的 DataTable 对象实现查询数据操作，则需要设置该 DataAdapter 对象的(　　)属性。

A. DeleteCommand　　B. InsertCommand
C. SelectCommand　　D. UpdateCommand

(19) 如果用一个 DataAdapter 对象填充 DataSet 对象，再使用该 DataSet 中的 DataTable 对象实现更新数据操作，则需要设置该 DataAdapter 对象的(　　)属性。

A. DeleteCommand　　B. InsertCommand
C. SelectCommand　　D. UpdateCommand

(20) 如果用一个 DataAdapter 对象填充 DataSet 对象，再使用该 DataSet 中的 DataTable 对象实现删除数据操作，则需要设置该 DataAdapter 对象的(　　)属性。

A. DeleteCommand　　B. InsertCommand
C. SelectCommand　　D. UpdateCommand

(21) 如果用一个 DataAdapter 对象填充 DataSet 对象，再使用该 DataSet 中的 DataTable 对象实现插入数据操作，则需要设置该 DataAdapter 对象的(　　)属性。

A. DeleteCommand　　B. InsertCommand
C. SelectCommand　　D. UpdateCommand

(22) 为了在程序中使用 SQL .NET 数据提供程序，应在源程序中添加对(　　)命名空间的的引用。

A. System. Data　　B. System. Data. SqlClient
C. System. Data. OleDb　　D. System. Data. Odbc. dll

(23) 为了在程序中使用 DataSet 对象，应在源程序中添加对(　　)命名空间的的引用。

A. System. Data　　B. System. Data. SqlClient
C. System. Data. OleDb　　D. System. Data. Odbc. dll

(24) 以下叙述中正确的是(　　)。

A. 一个网页中可以建立多个 Connection 对象连接数据库
B. Connection 对象使用完毕后可以不关闭
C. 可以使用 new 关键字创建 DataReader 对象
D. 可以使用 DataReader 对象更新数据库

(25) 以下叙述中正确的是(　　)。

A. 一个 DataSet 对象中只能包含一个 DataTable 对象
B. 一个 Connection 对象可以打开多个 DataReader 对象
C. DataRow 对象的 Delete 方法可以直接将该 DataRow 在 DataSet 中删除
D. 以上都不对

2. 问答题

(1) 在 ADO. NT 访问 SQL Server 时为什么需要数据提供程序?

(2) ADO .NET 中读写数据库需要用到那些对象? 它们的作用都是什么?

(3) 简述创建 DataAdapter 对象的 4 种方式。

(4) 简述 DataReader 对象和 DataSet 对象的异同。

(5) ADO .NET 允许哪两种方式从数据库中检索数据?

10.6　上机实验题

在 CH10 网站中添加一个 Exp 网页,其设计界面如图 10.23 所示,用户单击“学生平均分”按钮时在 ListBox1 中显示所有学生的平均分,如图 10.24 所示,采用 SqlDataReader 对象求解。用户单击“课程平均分”按钮时在 ListBox1 中显示所有课程的平均分,采用 DataSet 对象求解。

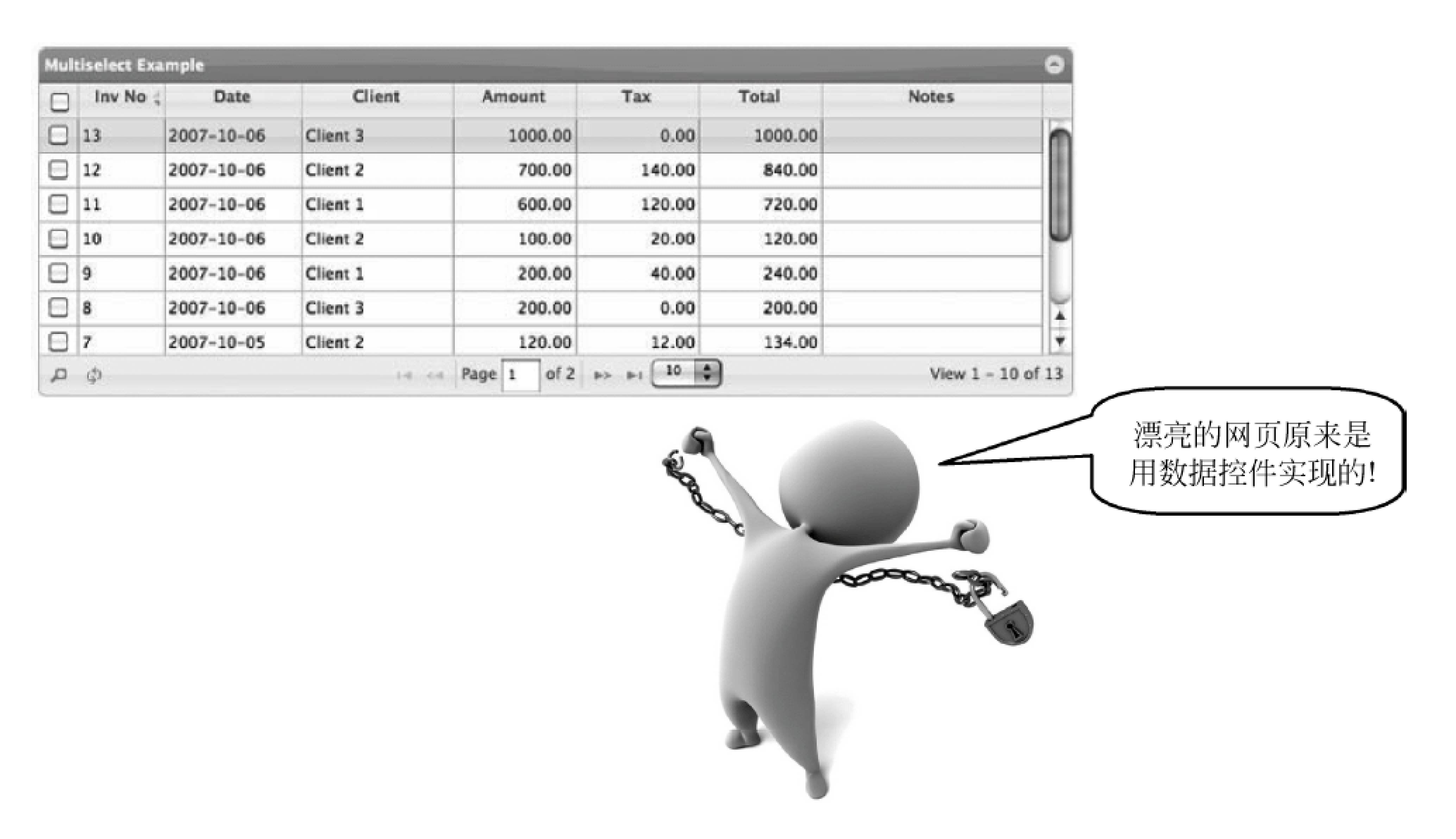


图 10.23　上机实验题网页的设计界面



图 10.24　上机实验题网页的执行界面

第 11 章　ASP .NET 数据控件



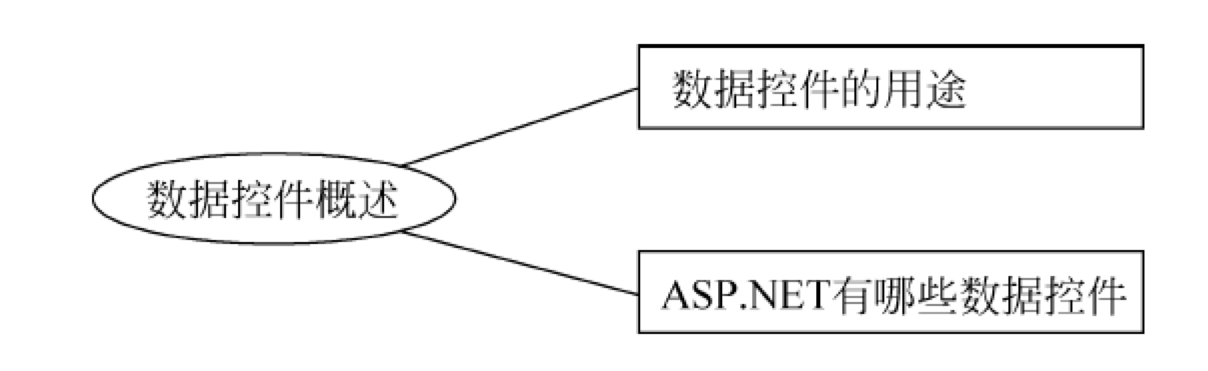
	Inv No	Date	Client	Amount	Tax	Total	Notes
	13	2007-10-06	Client 3	1000.00	0.00	1000.00	
	12	2007-10-06	Client 2	700.00	140.00	840.00	
	11	2007-10-06	Client 1	600.00	120.00	720.00	
	10	2007-10-06	Client 2	100.00	20.00	120.00	
	9	2007-10-06	Client 1	200.00	40.00	240.00	
	8	2007-10-06	Client 3	200.00	0.00	200.00	
	7	2007-10-05	Client 2	120.00	12.00	134.00	

本章指南



11.1　数据控件概述

知识梳理



11.1.1　数据控件的用途

数据控件分为数据源控件和数据绑定控件两种类型。

SQL Server 数据库中的数据与网页交互分为两个阶段：第 1 阶段是数据库与内存交互，包括将数据库中的数据提取到本地计算机的内存中，或将内存中的数据更改反映到数据库中；第 2 阶段是内存与网页交互，包括将内存在的数据中网页中显示，或用户操作网页更新内存中的数据。

对于第 1 阶段，第 10 章介绍了内存数据使用 DataSet 对象存储的编程方法，其思路是：SqlConnection 对象→SqlCommand 对象→SqlDataAdapter 对象→DataSet 对象，其编程过程比较繁琐。为了方便编程，ASP .NET 提供了一些数据源控件，封装上述过程的复杂性，直接实现从数据库提取数据到内存的整个过程。实际上，数据源控件允许使用不同类型的数据源，如数据库、XML 文件或中间层业务对象。数据源控件连接到数据源，从中检索数据，并使得其他控件可以绑定到数据源而无须代码，数据源控件还支持修改数据。

对于第 2 阶段，第 10 章介绍了 DataSet 对象中存储的数据在网页中显示的方法。同样，其编程过程比较繁琐。为了方便编程，ASP .NET 也提供了相应的数据绑定控件，以封装在网页中显示数据的复杂性。

数据控件的用途如图 11.1 所示，数据源控件实现数据库和内存数据的交互，只能检索和更新数据库的数据，不具有显示数据的能力。数据绑定控件实现内存数据和网页的交互，并具有在网页中显示数据的功能。

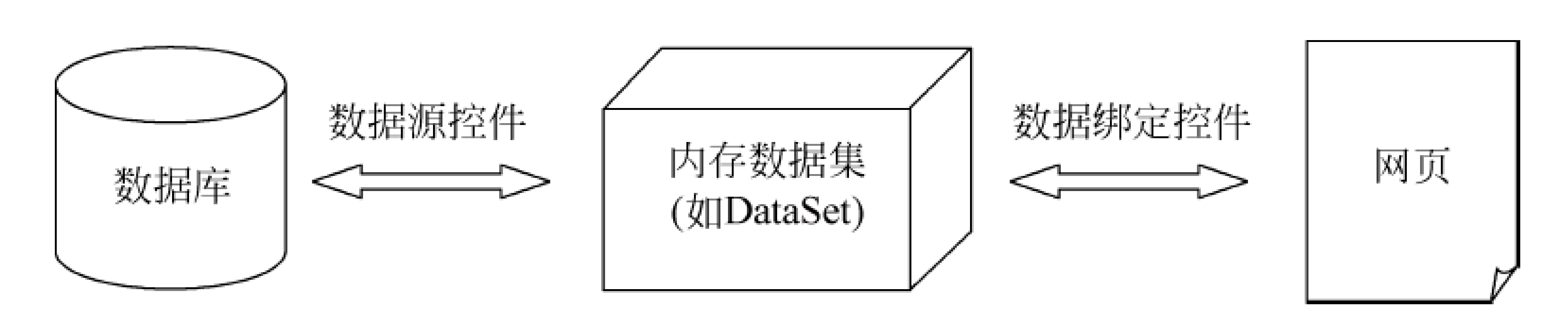


图 11.1　数据控件的用途

11.1.2　ASP .NET 有哪些数据控件

ASP .NET 工具箱的“数据”类别中的控件都是数据控件，如图 11.2 所示。每个控件具有不同的用途和所有环境。其中，数据源控件如表 11.1 所示，它们都是从 DataSourceControl 类派生的。

图 11.2 中的其他属于数据绑定控件，数据绑定就是把数据连接到网页的过程，在数据绑定后，可以通过网页界面来操作数据库中的数据。数据绑定控件将数据以标记的形式呈现给请求数据的浏览器。数据绑定控件可以绑定到数据源控件，并自动在页请求生命周期的适当时间获取数据。数据绑定控件可以利用数据源控件提供的功能，包括排序、分页、缓存、筛选、更新、删除和插入。数据绑定控件通过其 DataSourceID 属性连接到数据源控件。ASP .NET 中常见的数据绑定控件如表 11.2 所示。

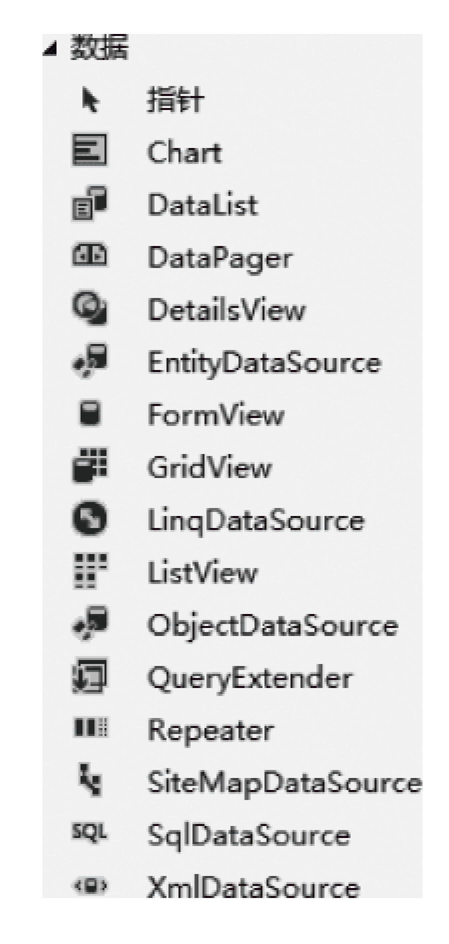


图 11.2　工具箱的“数据”类别的控件

表 11.1　常用的数据源控件及其说明

数据源控件	说　明
SqlDataSource	表示数据绑定控件的 SQL 数据库
ObjectDataSource	表示为多层 Web 应用程序体系结构中的数据绑定控件提供数据的业务对象
XmlDataSource	表示数据绑定控件的 XML 数据源
SiteMapDataSource	提供了一个数据源控件，Web 服务器控件及其他控件可使用该控件绑定到分层的站点地图数据
EntityDataSource	表示 ASP .NET 应用程序中数据绑定控件的实体数据模型(EDM)
LinqDataSource	支持通过标记文本在 ASP .NET 网页中使用语言集成查询(LINQ)，以在数据对象中检索和修改数据

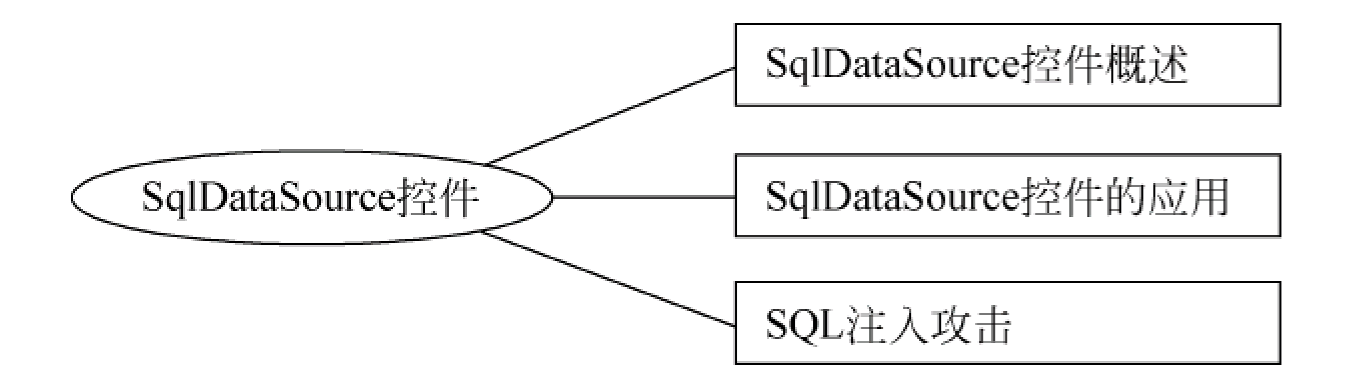
表 11.2　常见的数据绑定控件及其说明

数据绑定控件	说　明
列表控件	以各种列表形式呈现数据，列表控件包括 BulletedList、CheckBoxList、DropDownList、ListBox 和 RadioButtonList 控件
GridView	以表的形式显示数据，并支持在不编写代码的情况下对数据进行编辑、更新、排序和分页
DataList	以表的形式呈现数据，每一项都使用用户定义的项模板呈现
DetailsView	以表格布局一次显示一个记录，并允许编辑、删除和插入记录，还可以翻阅多个记录
FormView	与 DetailsView 控件类似，但允许用户为每一个记录定义一种自动格式的布局。对于单个记录，FormView 控件与 DataList 控件类似
AdRotator	将广告作为图像呈现在网页上，用户可以单击该图像来转到与广告关联的 URL
Menu	在可以包括子菜单的分层动态菜单中呈现数据
Repeater	以列表的形式呈现数据，每一项都使用用户定义的项模板呈现
TreeView	以可展开节点的分层树的形式呈现数据

本章主要介绍常用的几个数据源控件和数据绑定控件。

11.2　SqlDataSource 控件

知识梳理



11.2.1　SqlDataSource 控件概述

SqlDataSource 控件是最常用的数据源控件，允许使用 SQL Server、OLE DB、ODBC 或 Oracle 数据库。与 SQL Server 一起使用时支持高级缓存功能。当它的数据作为 DataSet 对象返回时，还支持排序、筛选和分页等功能。

SqlDataSource 控件对应的类为 SqlDataSource，它表示到数据库的直接连接。数据绑定控件(如 GridView、DetailsView 和 FormView 控件)可以使用 SqlDataSource 控件自动检索和

修改数据。可以将用来选择、插入、更新和删除数据的命令指定为 SqlDataSource 控件的一部分,并让该控件自动执行这些操作。用户无须编写代码(如使用 System. Data 命名空间中的类的 ADO .NET 代码)来创建连接并指定用于查询和更新数据库的命令。

SqlDataSource 控件的构造函数如下:

- SqlDataSource():初始化 SqlDataSource 类的新实例。
- SqlDataSource(string connectionString,string selectCommand):使用指定的连接字符串和 SELECT 命令初始化 SqlDataSource 类的新实例。
- SqlDataSource(string providerName,string connectionString,string selectCommand):使用指定的连接字符串和 Select 命令初始化 SqlDataSource 类的新实例。

其中,providerName 参数指出 SqlDataSource 使用的数据提供程序的名称,如果没有设置任何提供程序,则在默认情况下,SqlDataSource 使用 Microsoft SQL Server 的 ADO .NET 提供程序。connectionString 参数作为与基础数据库建立连接的连接字符串。selectCommand 参数用于从基础数据库中检索数据的 SQL 查询,如果该 SQL 查询是参数化的 SQL 字符串,可能需要将 Parameter 对象添加到 SelectParameters 集合中。

SqlDataSource 控件的常用属性、方法和事件分别如表 11.3～表 11.5 所示。

表 11.3　SqlDataSource 控件的常用属性及其说明

属性	说　明
ConnectionString	获取或设置特定于 ADO .NET 提供程序的连接字符串,SqlDataSource 控件使用该字符串连接基础数据库如 SQL Server 数据库
DataSourceMode	获取或设置 SqlDataSource 控件获取数据所用的数据检索模式,取值为 DataSet(默认值)或 DataReader
DeleteCommand	获取或设置 SqlDataSource 控件从基础数据库删除数据所用的 SQL 字符串
DeleteCommandType	获取或设置一个值,该值指示 DeleteCommand 属性中的文本是 SQL 语句还是存储过程的名称
DeleteParameters	集合属性,表示 SqlDataSource 控件 DeleteCommand 属性所使用参数的参数集合
FilterExpression	获取或设置调用 Select 方法时应用的筛选表达式
FilterParameters	集合属性,表示与 FilterExpression 字符串中的任何参数占位符关联参数的集合
InsertCommand	获取或设置 SqlDataSource 控件将数据插入基础数据库所用的 SQL 字符串
InsertCommandType	获取或设置一个值,该值指示 InsertCommand 属性中的文本是 SQL 语句还是存储过程的名称
InsertParameters	集合属性,表示 SqlDataSource 控件 InsertCommand 属性所使用的参数的参数集合
ProviderName	获取或设置 .NET Framework 数据提供程序的名称,SqlDataSource 控件使用该提供程序来连接基础数据源
SelectCommand	获取或设置 SqlDataSource 控件从基础数据库检索数据所用的 SQL 字符串
SelectCommandType	获取或设置一个值,该值指示 SelectCommand 属性中的文本是 SQL 查询还是存储过程的名称
SelectParameters	集合属性,表示 SqlDataSource 控件包含 SelectCommand 属性所使用的参数的参数集合
SortParameterName	获取或设置存储过程参数的名称,在使用存储过程执行数据检索时,该存储过程参数用于对检索到的数据进行排序
UpdateCommand	获取或设置 SqlDataSource 控件更新基础数据库中的数据所用的 SQL 字符串
UpdateCommandType	获取或设置一个值,该值指示 UpdateCommand 属性中的文本是 SQL 语句还是存储过程的名称
UpdateParameters	集合属性,表示 SqlDataSource 控件 UpdateCommand 属性所使用的参数的参数集合

表 11.4 SqlDataSource 控件的常用方法及其说明

方 法	说 明
DataBind	将数据源绑定到被调用的服务器控件及其所有子控件
Delete	使用 DeleteCommand 字符串和 DeleteParameters 集合中的所有参数执行删除操作
Insert	使用 InsertCommand 字符串和 InsertParameters 集合中的所有参数执行插入操作
Select	使用 SelectCommand 字符串以及 SelectParameters 集合中的所有参数从基础数据库中检索数据
Update	使用 UpdateCommand 字符串和 UpdateParameters 集合中的所有参数执行更新操作

表 11.5 SqlDataSource 控件的常用事件及其说明

事 件	说 明
DataBinding	当服务器控件绑定到数据源时引发
Deleted	完成删除操作后引发
Deleting	执行删除操作前引发
Disposed	当从内存释放服务器控件时引发，这是请求 ASP .NET 页时服务器控件生存期的最后阶段
Filtering	执行筛选操作前引发
Init	当服务器控件初始化时引发，初始化是控件生存期的第一步
Inserted	完成插入操作后引发
Inserting	执行插入操作前引发
Load	当服务器控件加载到 Page 对象中时引发
PreRender	在加载 Control 对象之后、呈现之前引发
Selected	完成数据检索操作后引发
Selecting	执行数据检索操作前引发
Unload	当服务器控件从内存中卸载时引发
Updated	完成更新操作后引发
Updating	执行更新操作前引发

SqlDataSource 控件提供了选择和显示数据，对数据进行排序、分页和缓存，更新、插入和删除数据，使用运行时参数筛选数据等。其主要的功能及说明如表 11.6 所示。

表 11.6 SqlDataSource 控件的功能及说明

功能	说 明
缓存	将 DataSourceMode 属性设置为 DataSet 值，EnableCaching 属性设置为 True，并根据希望缓存数据所具有的缓存行为设置 CacheDuration 和 CacheExpirationPolicy 属性
删除	将 DeleteCommand 属性设置为删除数据所用的 SQL 语句，此语句通常是参数化的
筛选	将 DataSourceMode 属性设置为 DataSet 值。将 FilterExpression 属性设置为在调用 Select 方法时用于筛选数据的筛选表达式
插入	将 InsertCommand 属性设置为插入数据所用的 SQL 语句，此语句通常是参数化的
分页	SqlDataSource 当前不支持此功能，但是将 DataSourceMode 属性设置为 DataSet 值时，某些数据绑定控件（如 GridView）支持分页
选择	将 SelectCommand 属性设置为检索数据所用的 SQL 语句
排序	将 DataSourceMode 属性设置为 DataSet
更新	将 UpdateCommand 属性设置为更新数据所用的 SQL 语句，此语句通常是参数化的

11.2.2 SqlDataSource 控件的应用

可以将 SqlDataSource 控件连接到 SQL Server 数据库，然后使用某些控件（如 GridView）来显示或编辑数据。

【练一练】 在 D 盘的电子商务目录中建立一个 CH11 的子目录，将其作为网站目录，设计一个 webform1 网页，其功能是说明 SqlDataSource 控件的使用方法。

其设计步骤如下：

① 打开 CH11 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH11”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform1.aspx，其他保持默认项，单击“添加”按钮。

② 从工具箱的“数据”类别中将 SqlDataSource 控件拖曳到网页上。展开“SqlDataSource 任务”列表，单击“配置数据源”将显示“配置数据源”向导。出现“选择您的数据连接”对话框，单击“新建连接”按钮，如图 11.3 所示。

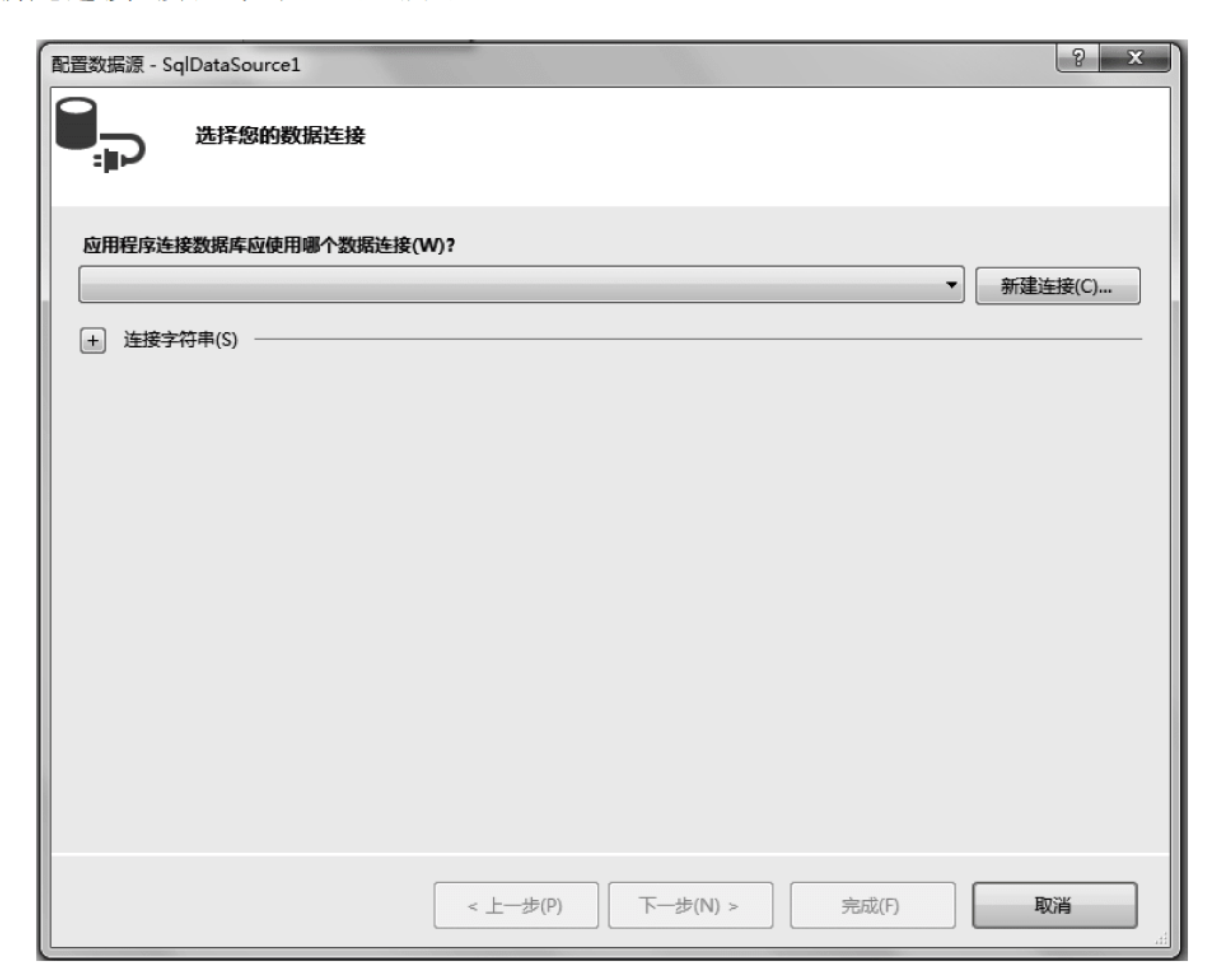


图 11.3 “选择您的数据连接”对话框

③ 出现如图 11.4 所示的“添加连接”对话框，单击“更改”按钮，在出现的对话框中选择 Microsoft SQL Server，单击“确定”按钮返回到“添加连接”对话框。

④ 在“服务器名”中输入或选择 LCB-PC 项，选择“使用 Windows 身份验证”单选按钮，选择 school 数据库，如图 11.5 所示。单击“测试连接”后出现连接成功信息，单击“确定”按钮返回。

⑤ 此时的新建连接名称为 lcb-pc.school.dbo。单击“下一步”按钮，出现“将连接字符串保存到应用程序配置文件中”对话框，保持默认值，即 web.config 文件中自动创建名称为 schoolConnectionString 的连接字符串，单击“下一步”按钮。



图 11.4　“添加连接”对话框一



图 11.5　“添加连接”对话框二

⑥ 出现"配置Select语句"对话框,选择student表,并选中所有列的复选框,如图11.6所示。单击"高级"按钮,出现"高级SQL生成选项"对话框,选中"生成INSERT、UPDATE和DELETE语句来更新数据源"复选框,如图11.7所示,单击"确定"按钮。



图11.6 "配置Select语句"对话框



图11.7 "高级SQL生成选项"对话框

⑦ 单击"下一步"按钮,在出现的对话框中单击"测试查询"按钮,出现如图11.8所示的页面,表示查找成功。单击"完成"按钮。

这样就在网页中建立好了SqlDataSource1控件,它对应的源视图代码如下:

```
<asp:SqlDataSource ID = "SqlDataSource1" runat = "server"
```

```
        ConnectionString = "<% $ ConnectionStrings:schoolConnectionString %>"
        DeleteCommand = "DELETE FROM [student] WHERE [学号] = @学号"
        InsertCommand = "INSERT INTO [student] ([姓名], [学号], [性别], [民族], [班号])
            VALUES (@姓名, @学号, @性别, @民族, @班号)"
        SelectCommand = "SELECT [姓名], [学号], [性别], [民族], [班号] FROM [student]"
        UpdateCommand = "UPDATE [student] SET [姓名] = @姓名, [性别] = @性别,
          [民族] = @民族, [班号] = @班号 WHERE [学号] = @学号">
        <DeleteParameters>
            <asp:Parameter Name = "学号" Type = "Int32" />
        </DeleteParameters>
        <InsertParameters>
            <asp:Parameter Name = "姓名" Type = "String" />
            <asp:Parameter Name = "学号" Type = "Int32" />
            <asp:Parameter Name = "性别" Type = "String" />
            <asp:Parameter Name = "民族" Type = "String" />
            <asp:Parameter Name = "班号" Type = "String" />
        </InsertParameters>
        <UpdateParameters>
            <asp:Parameter Name = "姓名" Type = "String" />
            <asp:Parameter Name = "性别" Type = "String" />
            <asp:Parameter Name = "民族" Type = "String" />
            <asp:Parameter Name = "班号" Type = "String" />
            <asp:Parameter Name = "学号" Type = "Int32" />
        </UpdateParameters>
    </asp:SqlDataSource>
```



图 11.8　"测试查询"页面

从上述代码看到，通过操作为 SqlDataSource1 控件设置了执行查询、插入、修改和删除的 SQL 语句，并以 student 表的主键"学号"为参数。

另外，在步骤⑤中选中“是否将连接保存到应用程序配置文件中”复选框，这样在 web. config 文件的中自动添加如下配置信息：

```
<connectionStrings>
    <add name="schoolConnectionString" connectionString="Data Source=LCB-PC;
        Initial Catalog=school;Integrated Security=True"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
```

选择“视图|服务器资源管理器”菜单命令，看到“服务器资源管理器”对话框如图 11.9 所示，表明在网站中建立了 schoolConnectionString 数据连接，网站的各个网页都是可以使用该数据连接。

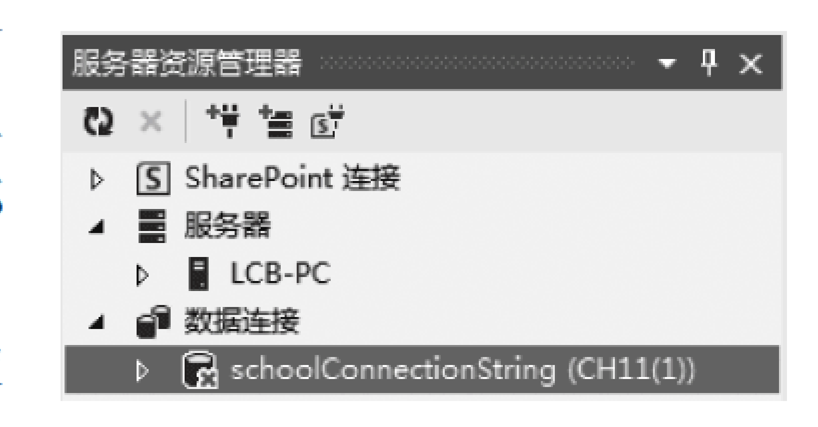


图 11.9　“服务器资源管理器”对话框

⑧ 由于 SqlDataSource 控件只能用于特定的数据绑定控件(如 GridView 控件等)以显示数据，这里在网页中拖曳一个 GridView 控件 GridView1，单击 GridView1 控件右上方的智能标记▶，在出现“GridView 任务”列表中，将“选择数据源”选择为 SqlDataSource1，并选中“启动分页”复选框，如图 11.10 所示，设置其 PageSize 属性为 3。

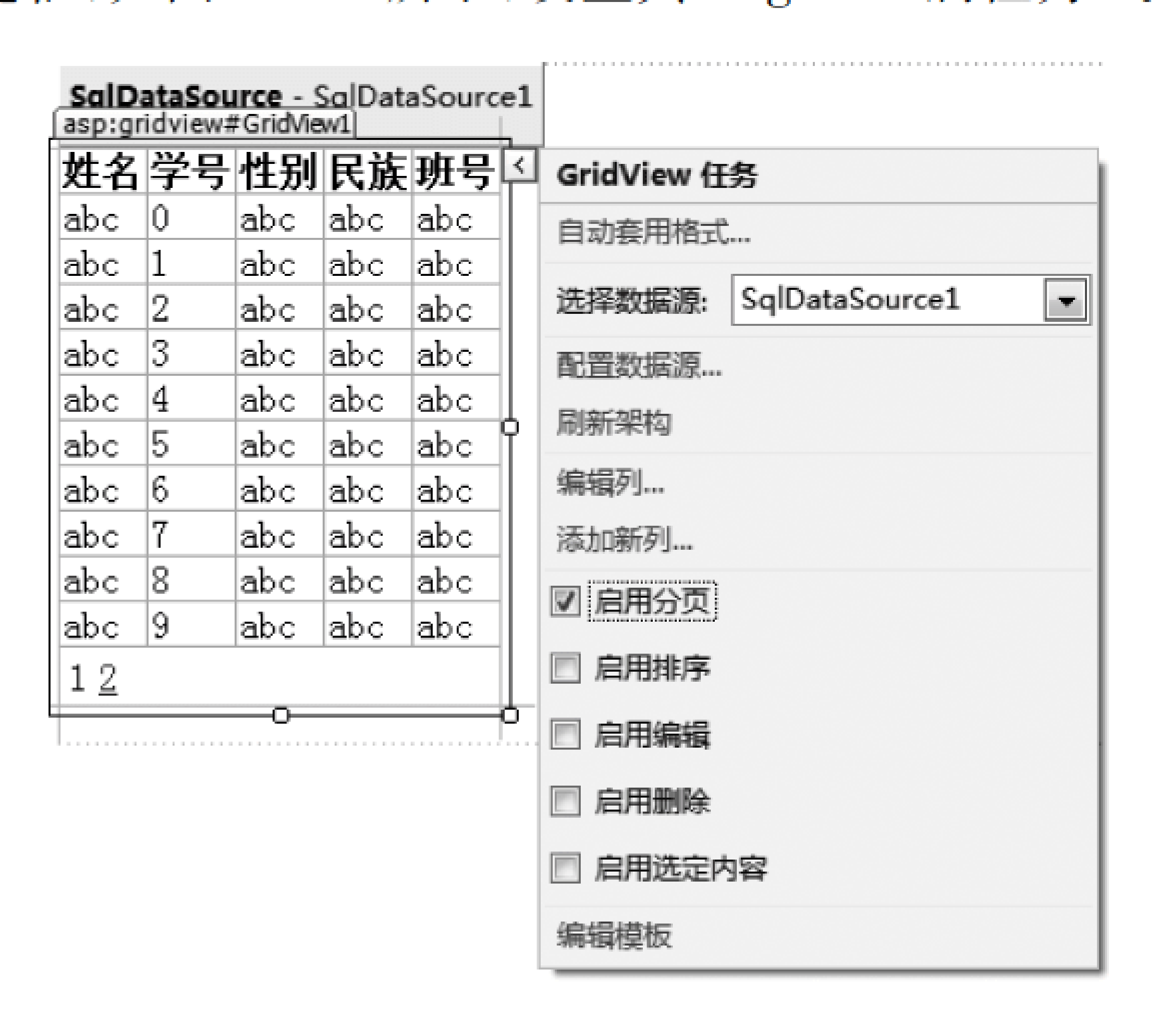


图 11.10　“GridView 任务”列表

说明：GridView 控件将在本章后面详细介绍，这里仅仅使用 GridView 控件显示数据的基本功能。

⑨ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，执行结果如图 11.11 所示，用户可以单击其他页号来显示对应页的数据。

从上例看到，SqlDataSource1 控件的 DeleteCommand、InsertCommand、SelectCommand 和 UpdateCommand 属性都是自动设置的，实际上，开发人员也可以定义这些属性和参数。

【练一练】　在本章 CH11 网站中添加一个 webform2 网页，其功能是采用 SqlDataSource 控件显示指定课程号的学生成绩表。

其设计步骤如下：

① 打开 CH11 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH11”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform2. aspx，其他保持默认项，单击“添加”

按钮。

② 本网页的设计界面如图 11.12 所示，其中有一个文本框 TextBox1、一个命令按钮 Button1、一个 SqlDataSource1 控件和一个 GridView1 控件。



图 11.11　webform1 网页的执行界面

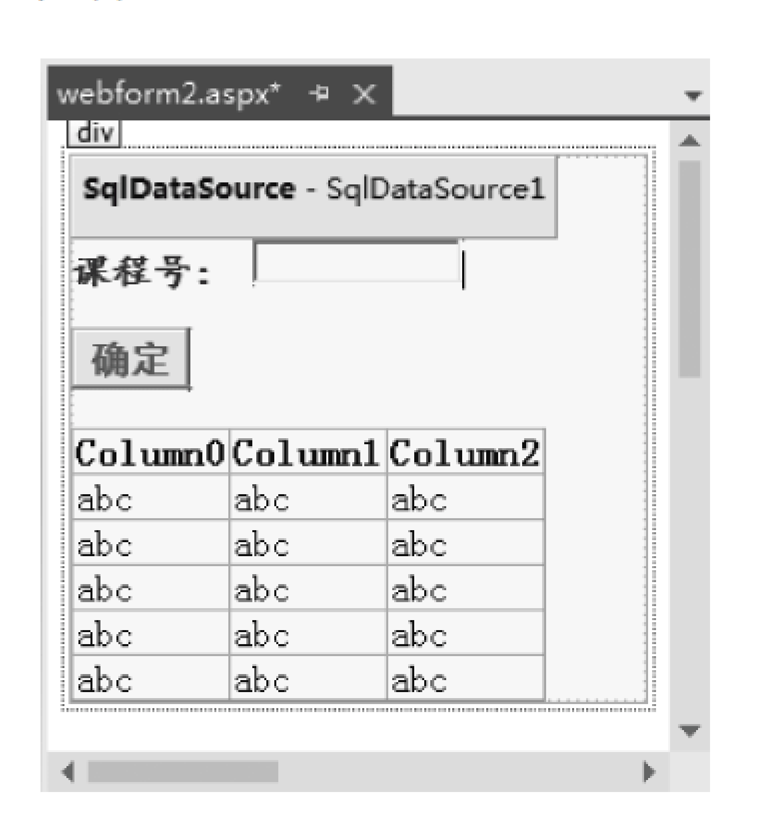


图 11.12　webform2 网页的设计界面

③ 展开 SqlDataSource1 控件的“SqlDataSource 任务”列表，选择“配置数据源”，在出现的“选择您的数据连接”页面中选择上例创建的 schoolConnectionString 数据连接，如图 11.13 所示。单击“下一步”按钮。



图 11.13　指定数据连接为 schoolConnectionString

④ 在出现的“配置 Select 语句”页面中，选择“指定自定义 SQL 语句或存储过程”，单击“下一步”按钮。

⑤ 在出现的“定义自定义语句或存储过程”页面中，选中 SELECT 选项卡，输入 SQL 语句如下：

```
SELECT student.学号,student.姓名,score.分数
FROM student,score
WHERE student.学号 = score.学号 AND score.课程号 = @kch
```

如图 11.14 所示，@kch 是定义的参数。单击“下一步”按钮。



图 11.14 “定义自定义语句或存储过程”页面

⑥ 出现“定义参数”页面，需要定制@kch 参数，因为该参数的值来源于网页中的 TextBox1 控件，所以从参数源下拉列表中选择 Control，ControlID 下拉列表中选择 TextBox1，指定默认值为 201，如图 11.15 所示。单击“下一步”按钮。

⑦ 出现“测试查询”页面，单击“完成”按钮。此时看到 SqlDataSource1 控件的源视图代码如下：

```
<asp:SqlDataSource ID = "SqlDataSource1" runat = "server"
    ConnectionString = "<% $ ConnectionStrings:schoolConnectionString %>"
    SelectCommand = "SELECT student.学号,student.姓名,score.分数
       FROM student,score
       WHERE student.学号 = score.学号 AND score.课程号 = @kch">
    <SelectParameters>
      <asp:ControlParameter ControlID = "TextBox1" DefaultValue = "201"
          Name = "kch" PropertyName = "Text" />
    </SelectParameters>
</asp:SqlDataSource>
```

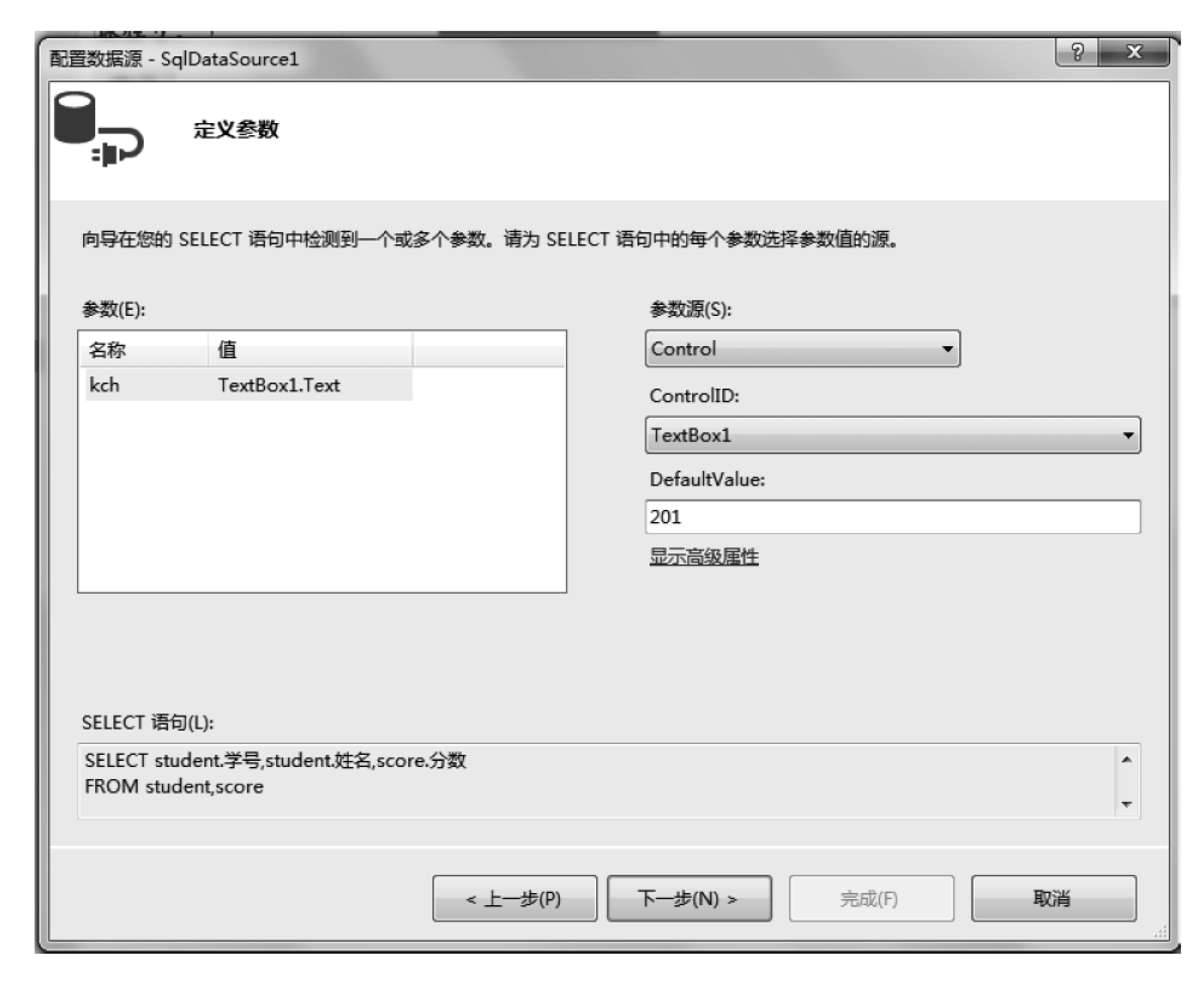


图 11.15　“定义参数”页面

从中看到，上述操作为 SqlDataSource1 控件设置了 SelectCommand 属性和相应的参数属性。实际上，如果开发人员熟悉 SqlDataSource 控件，也可以不使用向导而直接用代码创建 SqlDataSource 控件。

⑧ 单击 GridView1 控件右上方的智能标记，在出现的“GridView 任务”列表中，将“选择数据源”选择为 SqlDataSource1。

⑨ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，其初始执行界面如图 11.16 所示，显示 201 课程的学生成绩。输入课程号 101，单击“确定”按钮，其执行结果如图 11.17 所示。



图 11.16　webform2 网页的执行界面一



图 11.17　webform2 网页的执行界面二

本例的命令按钮 Button1 上没有设计任何自定义的代码,它仅仅取得提交的作用。在网页提交时,SqlDataSource1 控件自动更新数据并在 GridView1 控件中显示。

11.2.3 SQL 注入攻击

SQL 注入是开发人员未预期地把 SQL 代码传入到应用程序的过程。只有直接使用用户提供的值构建 SQL 语句的应用程序才会受到影响,因为此时用户可以输入特殊的来查看所有数据,这就是 SQL 注入攻击。

例如,在 CH11 网站中添加一个 webform2-1 的网页,其设计界面与 webform2 网页完完全相同。但没有使用 SqlDataSource 控件,而是直接使用用户提供的值构建 SELECT 语句,在该网页中设计如下事件处理方法:

```
protected void Button1_Click(object sender, EventArgs e)
{   if (TextBox1.Text != "")
    {   string mystr, mysql;
        SqlConnection myconn = new SqlConnection();
        SqlCommand mycmd = new SqlCommand();
        mystr = System.Configuration.ConfigurationManager.
            ConnectionStrings["schoolConnectionString"].ToString();
        myconn.ConnectionString = mystr;
        myconn.Open();
        mysql = "SELECT student.学号,student.姓名,score.分数 ";
        mysql += "FROM student,score ";
        mysql += "WHERE student.学号 = score.学号 AND score.课程号 = '"
            + TextBox1.Text.Trim() + "'";
        DataSet myds = new DataSet();
        SqlDataAdapter myda = new SqlDataAdapter(mysql, myconn);
        myda.Fill(myds, "mydata");
        myconn.Close();
        GridView1.DataSource = myds.Tables["mydata"];
        GridView1.DataBind();
    }
}
```

执行 webform2-1 网页,在课程号文本框中输入“aa' OR '1'='1”(SQL 注入攻击字符串),其结果如图 11.18 所示,这是因为此时构成的 SELECT 语句如下:

```
SELECT student.学号,student.姓名,score.分数
FROM student,score
WHERE student.学号 = score.学号 AND score.课程号 = 'aa' OR '1' = '1'
```

尽管上面语句结果没有意义,但在许多情况下会获取有意义的信息。解决的方法之一是将单引号替换成两个单引号,这样就不会和 SQL 语句中的分隔符混淆,如将 TextBox1.Text.Trim()用 TextBox1.Text.Trim().Replace("'","''") + "'"表示。当然,如果文本框中确实需要包含单引号,这样做也会出现问题。

SqlDataSource 控件考虑了这种 SQL 注入攻击。例如,执行 webform2 网页,输入同样的 SQL 注入攻击字符串,其结果如图 11.19 所示,未显示任何记录说明没有出现 SQL 注入攻击。



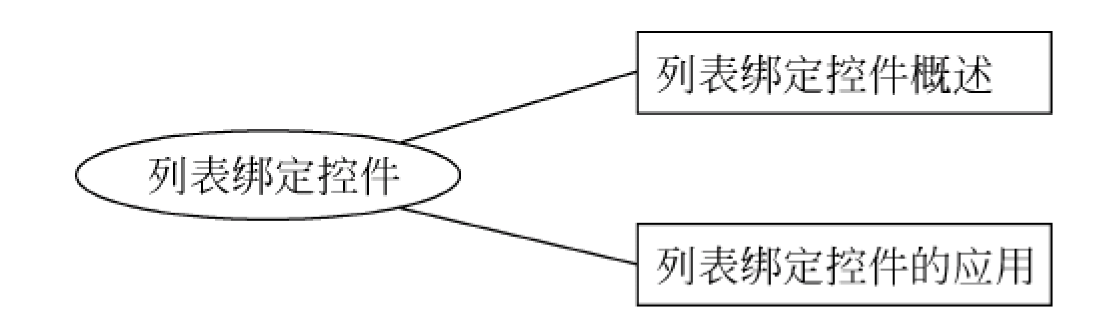
图 11.18　webform2-1 网页的执行界面



图 11.19　webform2 网页的执行界面

11.3　列表绑定控件

知识梳理



11.3.1　列表绑定控件概述

常用的列表绑定控件有 DropDownList、ListBox 控件等，它们的基本使用方法已在第 7 章介绍，这里主要讨论这些列表控件如何与数据库表进行数据绑定。

在实现数据绑定时，列表控件主要需要设置 DataSource(数据源)和 DataTextField(显示的绑定字段)属性，然后调用 DataBind 方法。

例如，如下代码通过数据绑定，在下拉列表 DropDownList1 控件中显示所有不相同的班号，以便用户从中选择一个班号：

```
string mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myconnstring"].ToString();
SqlConnection myconn = new SqlConnection();
myconn.ConnectionString = mystr;
myconn.Open();
DataSet myds = new DataSet();
SqlDataAdapter myda = new SqlDataAdapter("SELECT distinct 班号 FROM student",myconn);
myda.Fill(myds, "student");
DropDownList1.DataSource = myds.Tables["student"];
DropDownList1.DataTextField = "班号";
DropDownList1.DataBind();            //数据绑定
myconn.Close();
```

如果 DropDownList1 控件绑定的是 SqlDataSource1 控件，只需设置 DataSourceID 属性为 SqlDataSource1，设置 DataTextField 属性为 SqlDataSource1 控件中 SELECT 语句的字段列表。

11.3.2　列表绑定控件的应用

【练一练】 在本章 CH11 网站中添加一个 webform3 网页，其功能是说明列表控件的使用方法。

其设计步骤如下：

① 打开 CH11 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH11”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform3.aspx，其他保持默认项，单击“添加”按钮。

② 本网页的设计界面如图 11.20 所示，其中有一个 DropDownList1 控件、一个 ListBox1 控件、一个 Button1 控件和两个 SqlDataSource 控件（SqlDataSource1 和 SqlDataSource2）。



图 11.20　webform3 网页的设计界面

③ 通过操作设置所有控件的相关属性，本网页对应的源视图代码如下：

```
<%@ Page Language = "C#" AutoEventWireup = "true" CodeFile = "webform3.aspx.cs"
    Inherits = "webform3" %>
<!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head runat = "server">
    <meta http-equiv = "Content-Type" content = "text/html; charset = utf-8"/>
    <title></title>
    <style type = "text/css">
        .auto-style1 {
            font-family: 楷体; font-weight: bold;
            font-size: medium; color: #0000FF;
        }
    </style>
  </head>
  <body>
    <form id = "form1" runat = "server">
      <div class = "auto-style1">
        <span class = "auto-style1">班号:
        </span><asp:DropDownList ID = "DropDownList1"
           runat = "server" DataSourceID = "SqlDataSource1" DataTextField = "班号"
           Height = "24px" Width = "84px">
        </asp:DropDownList>
        <asp:SqlDataSource ID = "SqlDataSource1" runat = "server"
           ConnectionString = "<%$ ConnectionStrings:schoolConnectionString %>"
           SelectCommand = "SELECT distinct 班号 FROM student">
        </asp:SqlDataSource>
        <br /><br />
        <asp:Button ID = "Button1" runat = "server" style = "color: #FF0000; font-size:
```

```
          medium; font - weight: 700; font - family: 黑体" Text = "确定" />
     <br /><br />
     满足条件的记录<br />
   </div>
   <asp:ListBox ID = "ListBox1" runat = "server" DataTextField = "bf"
      DataSourceID = "SqlDataSource2" Width = "220px"
      style = "color:#FF00FF;font - size:medium;font - weight:700;font - family:仿宋">
   </asp:ListBox>
   <asp:SqlDataSource ID = "SqlDataSource2" runat = "server"
     ConnectionString = "<% $ ConnectionStrings:schoolConnectionString %>"
     SelectCommand = "SELECT CAST(学号 AS varchar) + ' ' + 姓名 + ' ' + 性别 + ' ' + 民族
        AS bf FROM student WHERE 班号 = @bh">
     <SelectParameters>
       <asp:ControlParameter ControlID = "DropDownList1" Name = "bh"
          PropertyName = "SelectedValue" />
     </SelectParameters>
   </asp:SqlDataSource>
  </form>
 </body>
</html>
```

④ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，其初始执行界面如图 11.21 所示，显示 15001 班的学生记录。在下拉列表中选择 15002，单击“确定”按钮，其执行结果如图 11.22 所示。



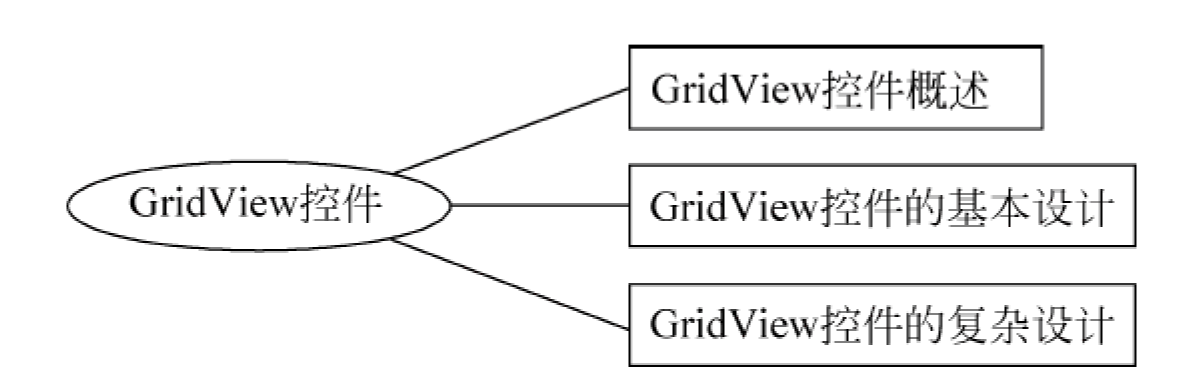
图 11.21　webform3 网页的执行界面一



图 11.22　webform3 网页的执行界面二

11.4　GridView 控件

知识梳理



11.4.1　GridView 控件概述

GridView 控件称为列表视图控件，是最重要的也最复杂的数据绑定控件之一。该控件用于在一个列表中显示数据源的值，其中每列表示一个字段，每行表示一条记录。它允许用户选择和编辑这些项以及对它们进行排序等。

GridView 控件的常用属性及其说明如表 11.7 所示，常用方法及其说明如表 11.8 所示，常用事件及其说明如表 11.9 所示。

表 11.7　GridView 控件的常用属性及其说明

属　性	说　明
AllowPaging	获取或设置一个值，该值指示是否启用分页功能
AllowSorting	获取或设置一个值，该值指示是否启用排序功能
AlternatingRowStyle	可以设置 GridView 控件中交替数据行的外观
AutoGenerateColumns	获取或设置一个值，该值指示是否为数据源中的每个字段自动创建绑定字段
Columns	获取表示 GridView 控件中列字段 DataControlField 对象的集合
DataKeyNames	获取或设置一个数组，该数组包含了显示在 GridView 控件中项主键字段的名称
DataKeys	获取一个 DataKey 对象集合，这些对象表示 GridView 控件中每一行的数据键值
DataMember	当数据源包含多个不同的数据项列表时，获取或设置数据绑定控件绑定到的数据列表的名称
DataSource	获取或设置对象，数据绑定控件从该对象中检索其数据项列表
DataSourceID	获取或设置控件的 ID
EditIndex	获取或设置要编辑行的索引
EditRowStyle	可以设置 GridView 控件中为进行编辑而选中行的外观
GridLines	获取或设置 GridView 控件的网格线样式
PageCount	获取在 GridView 控件中显示数据源记录所需的页数
PageIndex	获取或设置当前显示页的索引
PagerSettings	获取对 PagerSettings 对象的引用，使用该对象可以设置 GridView 控件中页导航按钮的属性
PagerStyle	获取对 TableItemStyle 对象的引用，使用该对象可以设置 GridView 控件中页导航行的外观
PagerTemplate	获取或设置 GridView 控件中页导航行的自定义内容
PageSize	获取或设置 GridView 控件在每页上所显示记录的数目
Rows	获取表示 GridView 控件中数据行 GridViewRow 对象的集合
SelectedDataKey	获取 DataKey 对象，该对象包含 GridView 控件中选中行的数据键值
SelectedIndex	获取或设置 GridView 控件中选中行的索引
SelectedRow	获取对 GridViewRow 对象的引用，该对象表示控件中的选中行
SelectedRowStyle	可以设置 GridView 控件中选中行的外观
SelectedValue	获取 GridView 控件中选中行的数据键值
SortDirection	获取正在排序列的排序方向
SortExpression	获取与正在排序列关联的排序表达式

表 11.8　GridView 控件的常用方法及其说明

方　法	说　明
DataBind	将数据源绑定到 GridView 控件
DeleteRow	从数据源中删除位于指定索引位置的记录
Sort	根据指定的排序表达式和方向对 GridView 控件进行排序
UpdateRow	使用行的字段值更新位于指定行索引位置的记录

表 11.9　GridView 控件的常用方法及其说明

事　件	说　明
DataBinding	当服务器控件绑定到数据源时发生
DataBound	在服务器控件绑定到数据源后发生
PageIndexChanged	在单击某一页导航按钮时，但在 GridView 控件处理分页操作之后发生
PageIndexChanging	在单击某一页导航按钮时，但在 GridView 控件处理分页操作之前发生
RowCommand	当单击 GridView 控件中的按钮时发生
RowDataBound	在 GridView 控件中将数据行绑定到数据时发生
RowDeleted	在单击某一行的“删除”按钮时，但在 GridView 控件删除该行之后发生
RowDeleting	在单击某一行的“删除”按钮时，但在 GridView 控件删除该行之前发生
RowEditing	发生在单击某一行的“编辑”按钮以后，GridView 控件进入编辑模式之前
RowUpdated	发生在单击某一行的“更新”按钮，并且 GridView 控件对该行进行更新之后
RowUpdating	发生在单击某一行的“更新”按钮以后，GridView 控件对该行进行更新之前
SelectedIndexChanged	发生在单击某一行的“选择”按钮，GridView 控件对相应的选择操作进行处理之后
SelectedIndexChanging	发生在单击某一行的“选择”按钮以后，GridView 控件对相应的选择操作进行处理之前
Sorted	在单击用于列排序的链接时，但在 GridView 控件对相应的排序操作进行处理之后发生
Sorting	在单击用于列排序的链接时，但在 GridView 控件对相应的排序操作进行处理之前发生

11.4.2　GridView 控件的基本设计

1. 绑定到数据设计

GridView 控件可绑定到数据源控件（如 SqlDataSource 等），以及实现 System. Collections. IEnumerable 接口的任何数据源（如 System. Data. DataView、System. Collections. ArrayList 或 System. Collections. Hashtable 等）。使用以下方法之一将 GridView 控件绑定到适当的数据源类型：

- 若要绑定到某个数据源控件，需将 GridViewID 属性设置为该数据源控件的 ID 值（如 SqlDataSource1）。GridView 控件自动绑定到指定的数据源控件，并且可利用该数据源控件的功能来执行排序、更新、删除和分页功能。这是绑定到数据的首选方法。
- 若要绑定到某个实现 System. Collections. IEnumerable 接口的数据源，如 DataSet 对象中的某个表，需以编程方式将 GridView 控件的 DataSource 属性设置为该数据源，然后调用 DataBind 方法。当使用此方法时，GridView 控件不提供内置的排序、更新、删除和分页功能。需要使用适当的事件提供此功能。

也就是说 GridView 控件的 DataSource 和 DataSourceID 两个属性不能同时使用。下面的例子说明第 1 种方法，在前面 webform11-1 网页设计就是采用第 2 种方法。

【练一练】 在本章 CH11 网站中添加一个 webform4 网页，其功能是说明 GridView 控件

绑定到 SqlDataSource 控件的使用方法。

其设计步骤如下：

① 打开 CH11 网站，选择"网站|添加新项"菜单命令，出现"添加新项-CH11"对话框，在中间列表中选择"Web 窗体"，将文件名称改为 webform4. aspx，其他保持默认项，单击"添加"按钮。

② 向本网页中拖放一个 GridView 控件 GridView1。展开"GridView 任务"列表，选择"新建数据源"命令。

③ 出现如图 11.23 所示的"选择数据源类型"页面，选中"数据库"，保持默认的名称为 SqlDataSource1，单击"确定"按钮。开始创建一个 SqlDataSource 控件。

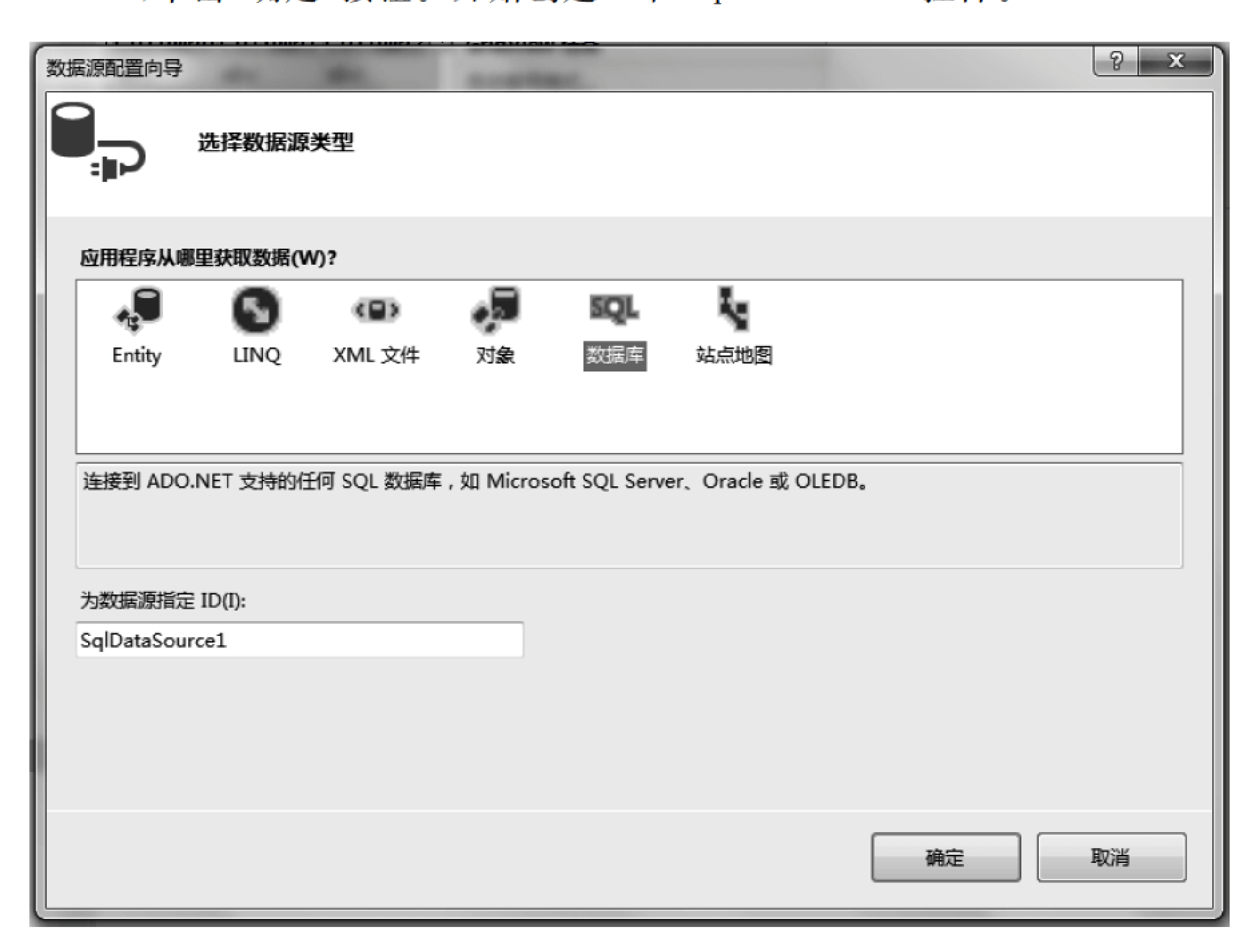


图 11.23 "选择数据源类型"页面

④ 在出现的"选择数据连接"对话框中选择前面创建的 schoolConnectionString 数据连接。单击"下一步"按钮。

⑤ 在出现的"配置 Select 语句"页面中，选择"指定自定义 SQL 语句或存储过程"，单击"下一步"按钮。

⑥ 在出现的"定义自定义语句或存储过程"页面中，输入如下 SELECT 语句：

```
SELECT student.学号,student.姓名,course.课程名,score.分数
FROM student,course,score
WHERE student.学号 = score.学号 AND course.课程号 = score.课程号
ORDER BY course.课程号,score.分数 DESC
```

如图 11.24 所示，单击"下一步"按钮。

⑦ 在出现的"测试查询"页面中，单击"完成"按钮。

⑧ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，其执行界面如图 11.25 所示，按课程号递增、分数递减列出所有学生的学号、姓名、课程名和分数信息。



图 11.24　"定义自定义语句或存储过程"页面

从本网页的源视图看到 GridView1 控件的代码如下：

```
<asp:GridView ID = "GridView1" runat = "server" AutoGenerate-
Columns = "False"
  DataKeyNames = "学号" DataSourceID = "SqlDataSource1">
  <Columns>
    <asp:BoundField DataField = "学号" HeaderText = "学号"
        ReadOnly = "True" SortExpression = "学号" />
    <asp:BoundField DataField = "姓名" HeaderText = "姓名"
        SortExpression = "姓名" />
    <asp:BoundField DataField = "课程名" HeaderText = "课程名"
        SortExpression = "课程名" />
    <asp:BoundField DataField = "分数" HeaderText = "分数"
        SortExpression = "分数" />
  </Columns>
</asp:GridView>
```



学号	姓名	课程名	分数
6	张军	C语言	90
8	马棋	C语言	88
1	王华	C语言	80
3	李兵	C语言	76
2	孙丽	C语言	70
6	张军	数据结构	92
1	王华	数据结构	83
8	马棋	数据结构	79
3	李兵	数据结构	70
2	孙丽	数据结构	52

图 11.25　webform4 网页的执行界面

也可以在网页在先创建好 SqlDataSourse 控件 SqlDataSourse1，然后设置 GridView1 控件的 DataSourceID 为 SqlDataSourse1。

2. 分页功能设计

如果需要分页，要将 AllowPaging 属性设置为 True，PageSize 属性设置一个页面中显示的记录个数，另外，PageCount 属性获取总的页数，PageIndex 获取当前的页号。PagerSettings 属性控制 GridView 控件中页导航行的设置，它是一个 PagerSettings 类对象，其常用的属性如表 11.10 所示。

表 11.10　PagerSettings 对象的常用属性及其说明

属性	说　　明
FirstPageImageUrl	获取或设置为第一页按钮显示的图像的 URL
FirstPageText	获取或设置为第一页按钮显示的文字
LastPageImageUrl	获取或设置为最后一页按钮显示的图像的 URL
LastPageText	获取或设置为最后一页按钮显示的文字
Mode	获取或设置支持分页的控件中的页导航控件的显示模式
NextPageImageUrl	获取或设置为下一页按钮显示的图像的 URL
NextPageText	获取或设置为下一页按钮显示的文字
PageButtonCount	获取或设置在 Mode 属性设置为 Numeric 或 NumericFirstLast 值时页导航中显示的页按钮的数量
Position	获取或设置一个值，该值指定页导航的显示位置
PreviousPageImageUrl	获取或设置为上一页按钮显示的图像的 URL
PreviousPageText	获取或设置为上一页按钮显示的文字

其中，PagerSettings 属性的 Mode 有 4 种取值：

- NextPrevious：上一页按钮和下一页按钮。
- NextPreviousFirstLast：上一页按钮、下一页按钮、第一页按钮和最后一页按钮。
- Numeric：可直接访问页面的带编号的链接按钮。
- NumericFirstLast：带编号的链接按钮、第一个链接按钮和最后一个链接按钮。

在 Mode 属性设置为 NextPrevious、NextPreviousFirstLast 或 NumericFirstLast 值时，可以通过设置 PagerSettings 属性的以下属性来自定义非数字按钮的文字：

- FirstPageText：第一页按钮的文字。
- PreviousPageText：上一页按钮的文字。
- NextPageText：下一页按钮的文字。
- LastPageText：最后一页按钮的文字。

也可以通过设置 PagerSettings 属性的以下属性为非数字按钮显示图像：

- FirstPageImageUrl：为第一页按钮显示图像的 URL。
- PreviousPageImageUrl：为上一页按钮显示图像的 URL。
- NextPageImageUrl：为下一页按钮显示图像的 URL。
- LastPageImageUrl：为最后一页按钮显示图像的 URL。

例如，对于 webform4 网页，将 GridView1 控件的 PageSize 改为 3，并设计 PagerSettings 属性如下：

```
<PagerSettings mode = "NextPreviousFirstLast"
    FirstPageText = "首页"
    NextPageText = "下一页"
    PreviousPageText = "上一页"
    LastPageText = "尾页"
    Position = "Bottom"/>
```

执行 webform4 网页时，各页的显示界面如图 11.26 所示。

3. 基本数据操作

GridView 控件提供了很多内置功能，这些功能使得用户可以对控件中的项进行排序、更







(b) 第2页



(c) 最后一页

图 11.26　各页的显示界面

新、删除、选择和分页。当 GridView 控件绑定到某个数据源控件时，GridView 控件可利用该数据源控件的功能并提供自动排序、更新和删除功能。

包含更新、删除或选择按钮或链接的列称为命令域。如图 11.27 所示，其中的 GridView1 控件用于操作 school 数据库的 course 表，并显示命令域(CommandField)。

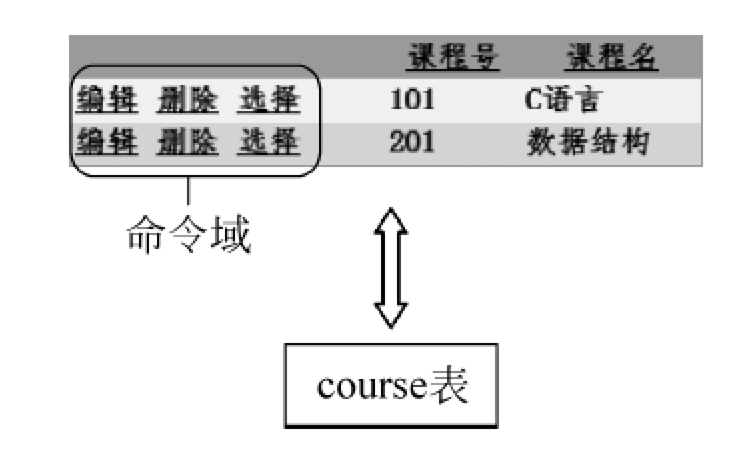


图 11.27　带有命令域的 GridView1 控件

(1) 删除操作

当用户单击命令域中的“删除”按钮时，网页被刷新，且对应的行将消失。删除操作是通过调用对应数据源控件的 Delete 方法(或对应的 DELETE 语句)实现的。

(2) 编辑操作

当用户单击“编辑”按钮时，网页被刷新，且对应的行进入可编辑状态，用户可以为可编辑字段输入新值。可编辑状态下的样式由 EditRowStyle 属性确定，此时，“编辑”按钮被“更新”和“取消”按钮代替，前者保存所做的修改，后者放弃所做的修改。

更新操作是通过调用对应数据源控件的 Update 方法(或对应的 UPDATE 语句)实现的。

(3) 选择操作

当用户单击“选择”按钮时，选中的行以 SelectedRowStyle 样式显示。SelectedRow 属性表示该行，SelectedIndex 属性表示该行的索引。

4. 其他重要的属性

(1) AutoGenerateColumns 属性

该属性获取或设置网页运行时是否基于关联的数据源自动生成列。其默认值为 True，也就是说，一旦指定了 GridView 控件的数据源，便自动生成相应的列。在有些情况下，不希望自动生成列，而是通过“GridView 任务”列表的“编辑列”命令来设置相关的列，此时要将 AutoGenerateColumns 属性设为 False。

(2) DataKeyNames 和 DataKeys 属性

DataKeyNames 属性是一个字符串数组，指定表示数据源主键的字段。当设置了 DataKeyNames 属性时，GridView 控件自动为该控件中的每一行创建一个 DataKey 对象。DataKey 对象包含在 DataKeyNames 属性中的指定的字段的值。DataKey 对象随后被添加到控件的 DataKeys 集合中。使用 DataKeys 属性检索 GridView 控件中特定数据行的 DataKey 对象。这提供了一种访问每个行主键的便捷方法。例如：

```
GridView1.DataKeyNames = new string[] {"学号"};
…
```

```
TextBox1.Text = GridView1.DataKeys[0].Value.ToString();
```

GridView 控件的 DataKeyNames 属性可被设置为绑定到 GridView 的数据的主键列名，如果设置了该属性，GridView 控件将自动跟踪每行的主键列值。当绑定数据源控件到 GridView 控件时，该属性自动被设置为数据源控件返回的主键列。

(3) Rows 属性

Rows 属性用来获取表示 GridView 控件中数据行 GridViewRow 对象的集合。

通过使用 GridViewRow 对象的 Cells 属性，可以访问一行的单独单元格。如果某个单元格包含其他控件，则通过使用单元格的 Controls 集合，可以从单元格检索控件。如果控件指定了 ID，还可以使用单元格的 FindControl 方法来查找该控件。

若要从 BoundField 字段列或自动生成的字段列检索字段值，需使用单元格的 Text 属性。若要从将字段值绑定到控件的其他字段列类型检索字段值，需先从相应的单元格检索控件，然后访问该控件的相应属性。

例如，以下代码显示第 2 行的姓名：

```
TextBox1.Text = GridView1.Rows[1].Cells[1].Text;  //第 2 个单元格为姓名，索引均从 0 开始
```

【练一练】 在本章 CH11 网站中添加一个 webform5 网页，其功能是说明 GridView 控件的分页、编辑和删除功能的使用方法。

其设计步骤如下：

① 打开 CH11 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH11”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform5. aspx，其他保持默认项，单击“添加”按钮。

② 向本网页中拖曳一个 GridView 控件 GridView1。展开“GridView 任务”列表，选择“新建数据源”命令。出现“选择数据源类型”页面，选中“数据库”，保持默认的名称为 SqlDataSource1，单击“确定”按钮。开始创建一个 SqlDataSource 控件。

③ 在出现的“选择数据连接”页面中选择前面创建的 schoolConnectionString 数据连接。单击“下一步”按钮。

④ 像图 11. 6 和图 11. 7 那样操作，最后单击“完成”按钮返回。

⑤ 此时看到 GridView 任务列表中出现“启动分页”等复选框，选中各个复选框如图 11. 28 所示。

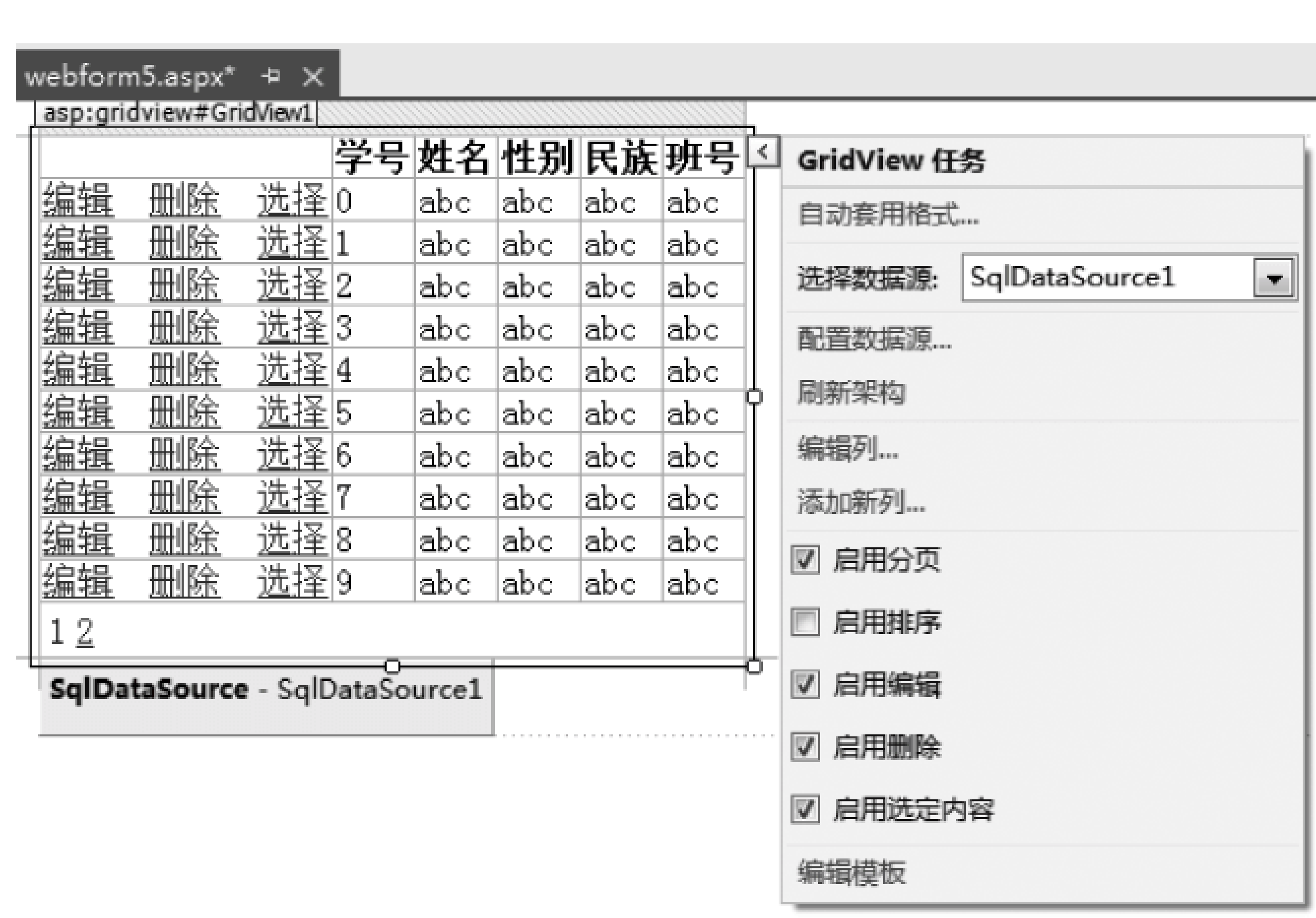


图 11. 28 勾选“启动分页”等

⑥ 将 GridView1 控件的 PageSize 属性设置为 3,并设置相应的字体和颜色属性。

⑦ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页,其执行界面如图 11.29 所示。用户可以单击“编辑”、“删除”或“选择”链接进行相应的操作。

图 11.29　webform5 网页的执行界面

webform5 网页中 GridView1 控件的源视图代码如下:

```
asp: GridView  ID  =  " GridView1 "  runat  =  " server "
AllowPaging = "True"
    AutoGenerateColumns = "False" DataKeyNames = "学号"
    DataSourceID = "SqlDataSource1" PageSize = "3" >
    < Columns >
        < asp:CommandField ShowDeleteButton = "True" ShowEditButton = "True"
            ShowSelectButton = "True" />
        < asp:BoundField DataField = "学号" HeaderText = "学号" ReadOnly = "True"
            SortExpression = "学号" />
        < asp:BoundField DataField = "姓名" HeaderText = "姓名" SortExpression = "姓名" />
        < asp:BoundField DataField = "性别" HeaderText = "性别" SortExpression = "性别" />
        < asp:BoundField DataField = "民族" HeaderText = "民族" SortExpression = "民族" />
        < asp:BoundField DataField = "班号" HeaderText = "班号" SortExpression = "班号" />
    </Columns >
</asp:GridView >
```

注意:本例没有编写一行代码,都是通过用户操作来实现网页设计的。通过查看源代码有几点说明如下:

① 自动设置 GridView1 控件的 SqlDataSource ID 属性为 SqlDataSource1。

② SqlDataSource1 控件用于操作 school 数据库的 student 表。

③ 当 GridView 控件绑定到数据源控件 SqlDataSource1 时,DataKeyNames 属性自动设置为数据源控件返回的主键列(这里为“学号”),GridView 控件中的每行都是以 DataKeyNames 属性值为关键字。

④ 如果不指定对应 SqlDataSource1 控件的 UpdateCommand、InsertCommand 和 DeleteCommand 属性,则 GridView1 控件不会支持更新、插入和删除操作。本例通过操作自动设置了这些属性。

11.4.3　GridView 控件的复杂设计

1. 列对象处理

(1) 列字段

前面的例子中,GridView 控件的各个列都是通过数据源控件自动生成的,也可以手工创建列。其操作是,选择“GridView 列表”中的“编辑列”命令,打开“字段”对话框,如图 11.30 所示,通过该对话框可进行列字段的添加、删除,或对列字段的属性进行设置等,如利用 HeaderText 属性设置列字段呈现的文本。

不同的列字段类型决定控件中各列的行为方式。表 11.11 列出了可以使用的不同列字段类型。

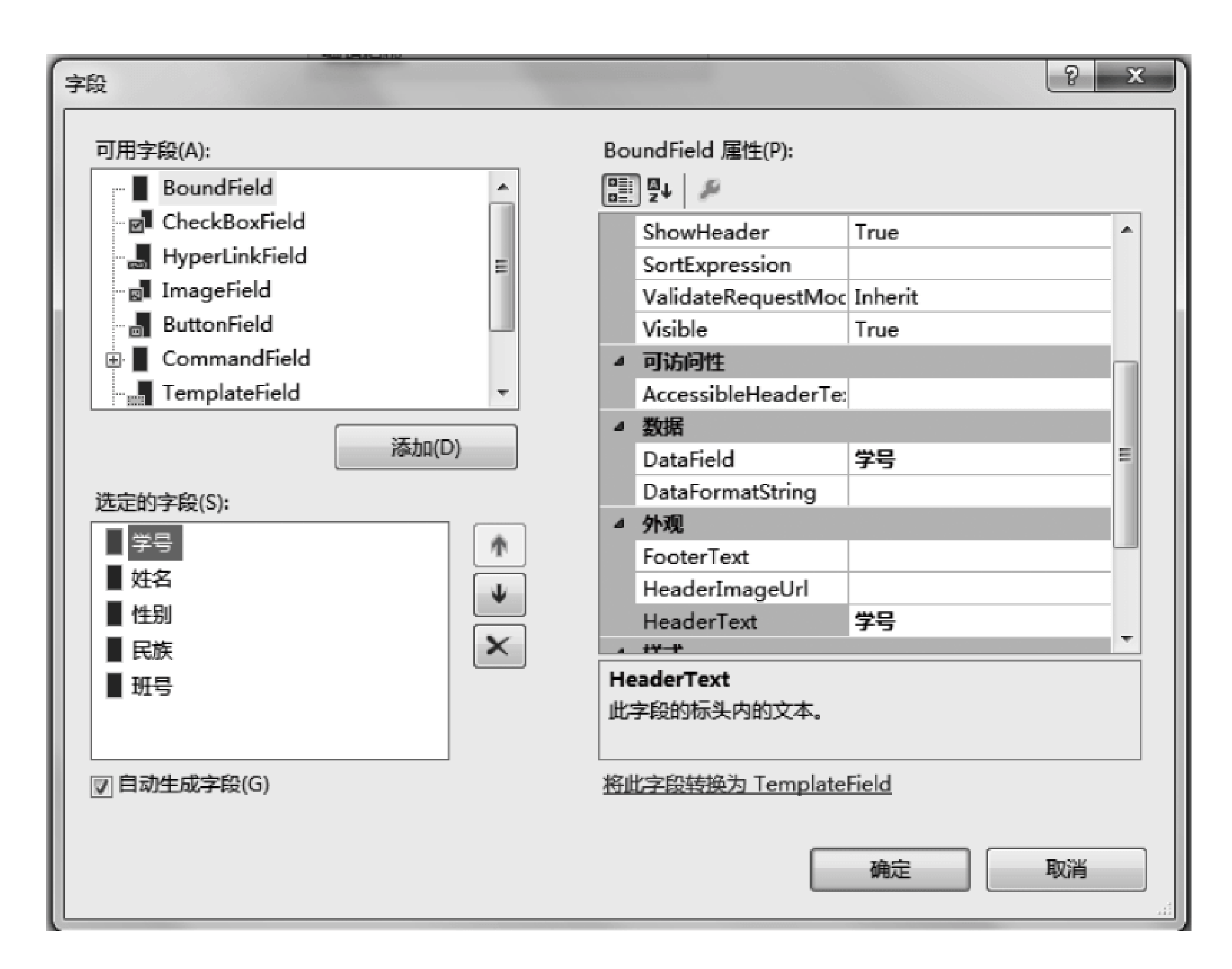


图 11.30　“字段”对话框

表 11.11　GridView 控件列字类型及其说明

列字段类型	说　　明
BoundField	显示数据源中某个字段的值，是 GridView 控件的默认列类型
ButtonField	为 GridView 控件中的命令域，其中每个项显示一个链接或按钮。这样可以创建一列自定义按钮控件，如“添加”或“移除”按钮
CheckBoxField	为 GridView 控件中的每一项显示一个复选框，此列字段类型通常用于显示具有布尔值的字段
CommandField	显示用来执行选择、编辑或删除操作的预定义命令按钮
HyperLinkField	将数据源中某个字段的值显示为超链接，此列字段类型允许将另一个字段绑定到超链接的 URL
ImageField	为 GridView 控件中的每一项显示一个图像
TemplateField	根据指定的模板为 GridView 控件中的每一项显示用户定义的内容，此列字段类型允许创建自定义的列字段

(2) CommandField 列字段

CommandField 列字段用于创建命令域，图 11.31 列出了 CommandField 列字段的属性，其中 ButtonType 用于设置其按钮类型，取值为 Link(默认值)时显示为链接，取值为 Button 时显示为按钮，取值为 Image 时显示为指定 URL 的图像。

CommandField 列字段几个重要的属性(均为布尔型)如下：

- ShowCancelButton：该字段用于设置是否向用户显示“取消”按钮。如果为 True，可用 CancelText 属性设置其标题。
- ShowDeleteButton：该字段用于设置是否向用户显示“删除”按钮。如果为 True，可用 DeleteText 属性设置其标题。

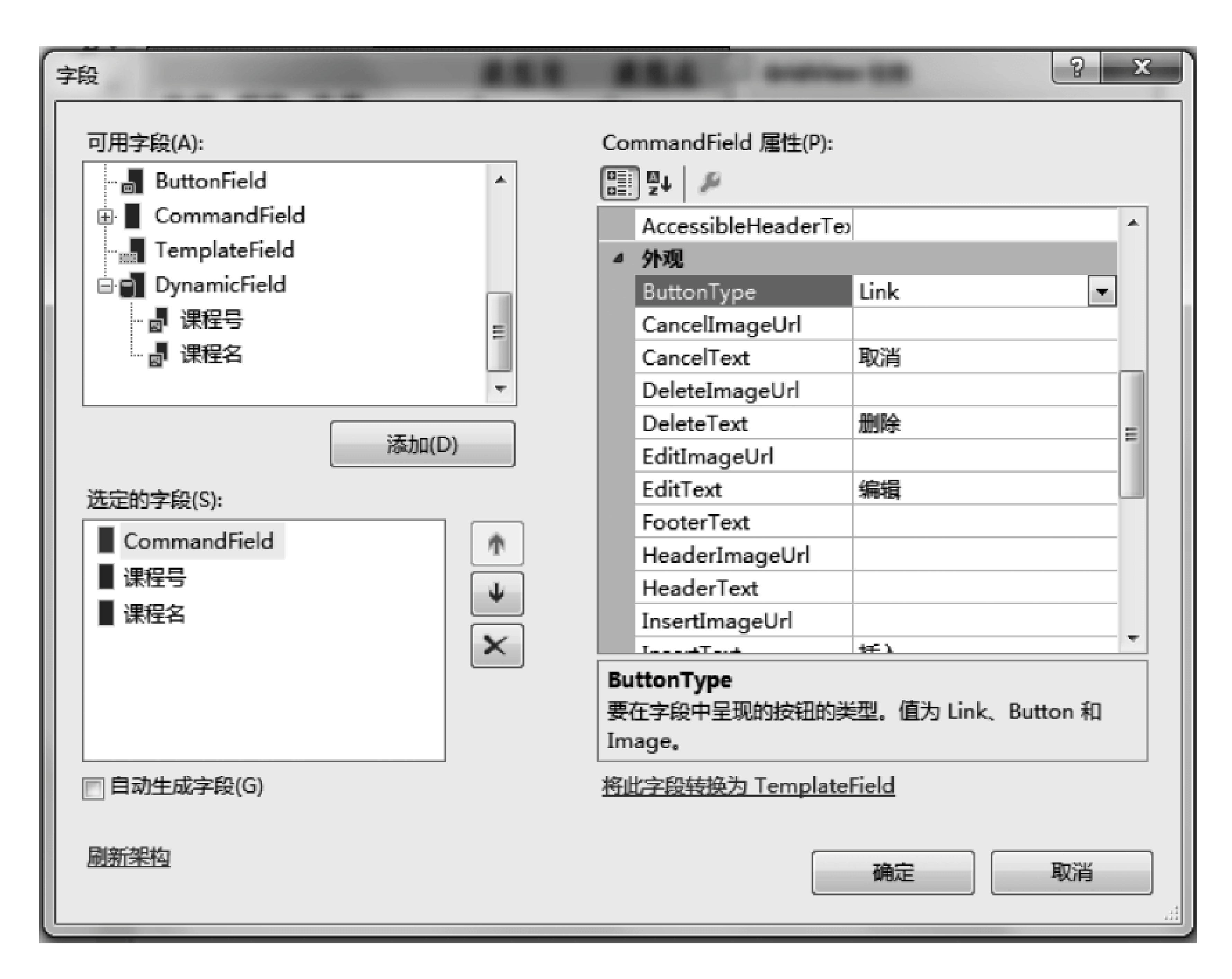
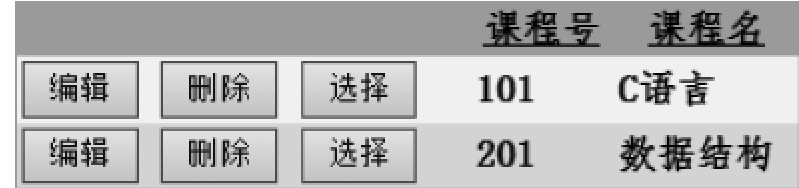


图 11.31　CommandField 列字段的属性

- ShowEditButton：该字段用于设置是否向用户显示“编辑”按钮。如果为 True，可用 EditText 属性设置其标题。
- ShowInsertButton：该字段用于设置是否向用户显示“插入”按钮。如果为 True，可用 InsertText 属性设置其标题。
- ShowSelectButton：该字段用于设置是否向用户显示“选择”按钮。如果为 True，可用 SelectText 属性设置其标题。

例如，将一个 CommandField 列字段的 ButtonType 设置为 Button，ShowDeleteButton、ShowEditButton 和 ShowSelectButton 设置为 True，其他为 False，其外观如图 11.32 所示。将一个 CommandField 列字段的 ButtonType 设置为 Button，ShowSelectButton 设置为 True，其他为 False，SelectText 设置为“选择一个记录”，其外观如图 11.33 所示。



			课程号	课程名
编辑	删除	选择	101	C语言
编辑	删除	选择	201	数据结构

图 11.32　CommandField 列字段的外观一

	课程号	课程名
选择一个记录	101	C语言
选择一个记录	201	数据结构

图 11.33　CommandField 列字段的外观二

(3) BoundField 列字段

BoundField 列字段用于创建绑定字段。GridView 控件中的一个列就是 Columns 集合属性的一个元素，该元素是 DataControlField 类对象。DataControlField 类的常见属性如表 11.12 所示。例如，如下语句设置 GridView1 控件第 1 列的标题为 Name：

```
GridView1.Columns[0].HeaderText = "Name";
```

表 11.12　DataControlField 类的常见属性及其说明

属　性	说　明
ControlStyle	获取 DataControlField 对象所包含的任何 Web 服务器控件的样式
FooterStyle	获取或设置数据控件字段脚注的样式
HeaderStyle	获取或设置数据控件字段标头的样式
HeaderText	获取或设置数据控件字段的标题项中显示的文本
ItemStyle	获取由数据控件字段显示的任何基于文本内容的样式
SortExpression	获取或设置数据源控件用来对数据进行排序的排序表达式

另外，像 HeaderStyle 这样的样式属性又是 TableItemStyle 类对象，具有如表 11.13 所示的常用属性。例如，如下语句设置 GridView1 控件第 1 列的标题宽度为 100 像素：

```
GridView1.Columns[0].HeaderStyle.Width = 100;
```

表 11.13　列的样式属性的常用属性及其说明

属　性	说　明
BorderColor	获取或设置 Web 服务器控件的边框颜色
BorderStyle	获取或设置 Web 服务器控件的边框样式
BorderWidth	获取或设置 Web 服务器控件的边框宽度
Font	获取与 Web 服务器控件关联的字体属性
ForeColor	获取或设置 Web 服务器控件的前景色(通常是文本颜色)
Height	获取或设置 Web 服务器控件的高度
HorizontalAlign	获取或设置单元格内容的水平对齐方式
VerticalAlign	获取或设置单元格内容的垂直对齐方式
Width	获取或设置 Web 服务器控件的宽度

另外，GridView 控件的 AlternatingRowStyle 属性可以设置其中的交替数据行的外观，SelectedRowStyle 属性可以设置其中选中行的外观。

例如，在 webform5 网页中设计如下设计处理过程，用于美化界面：

```
protected void Page_Load(object sender, EventArgs e)
{   int i;
    GridView1.AlternatingRowStyle.BackColor = System.Drawing.Color.Pink;
    GridView1.SelectedRowStyle.ForeColor = System.Drawing.Color.Red;
    GridView1.SelectedRowStyle.BackColor = System.Drawing.Color.DarkSeaGreen;
    GridView1.Columns[0].HeaderStyle.Width = 130;
    GridView1.Columns[0].ItemStyle.Font.Bold = true;
    GridView1.Columns[0].ItemStyle.Font.Name = "仿宋";
    GridView1.Columns[0].ItemStyle.Font.Size = 12;
    GridView1.Columns[0].ItemStyle.HorizontalAlign = HorizontalAlign.Center;
    GridView1.Columns[0].ItemStyle.ForeColor = System.Drawing.Color.CadetBlue;
    for (i = 1; i <= 5; i++)
    {   GridView1.Columns[i].HeaderStyle.Width = 70;
        GridView1.Columns[i].HeaderStyle.Font.Bold = true;
        GridView1.Columns[i].HeaderStyle.Font.Name = "黑体";
        GridView1.Columns[i].HeaderStyle.Font.Size = 14;
```

```
            GridView1.Columns[i].HeaderStyle.ForeColor = System.Drawing.Color.Red;
        }
        for (i = 1; i <= 5; i++)
        {   GridView1.Columns[i].ItemStyle.Font.Bold = true;
            GridView1.Columns[i].ItemStyle.Font.Name = "楷体";
            GridView1.Columns[i].ItemStyle.Font.Size = 12;
            GridView1.Columns[i].ItemStyle.HorizontalAlign = HorizontalAlign.Center;
            GridView1.Columns[i].ItemStyle.ForeColor = System.Drawing.Color.Black;
        }
    }
```

修改后 webform5 网页的执行界面如图 11.34 所示，看到标题字体和各行字体不同，同时设置了交替数据行的外观。单击第 3 行中“选择”链接，其界面如图 11.35 所示，看到选中行的外观也不相同。

	学号	姓名	性别	民族	班号
编辑 删除 选择	1	王华	女	汉族	15001
编辑 删除 选择	2	孙丽	女	满族	15002
编辑 删除 选择	3	李兵	男	汉族	15001
12					

图 11.34　修改后 webform5 网页的执行界面一

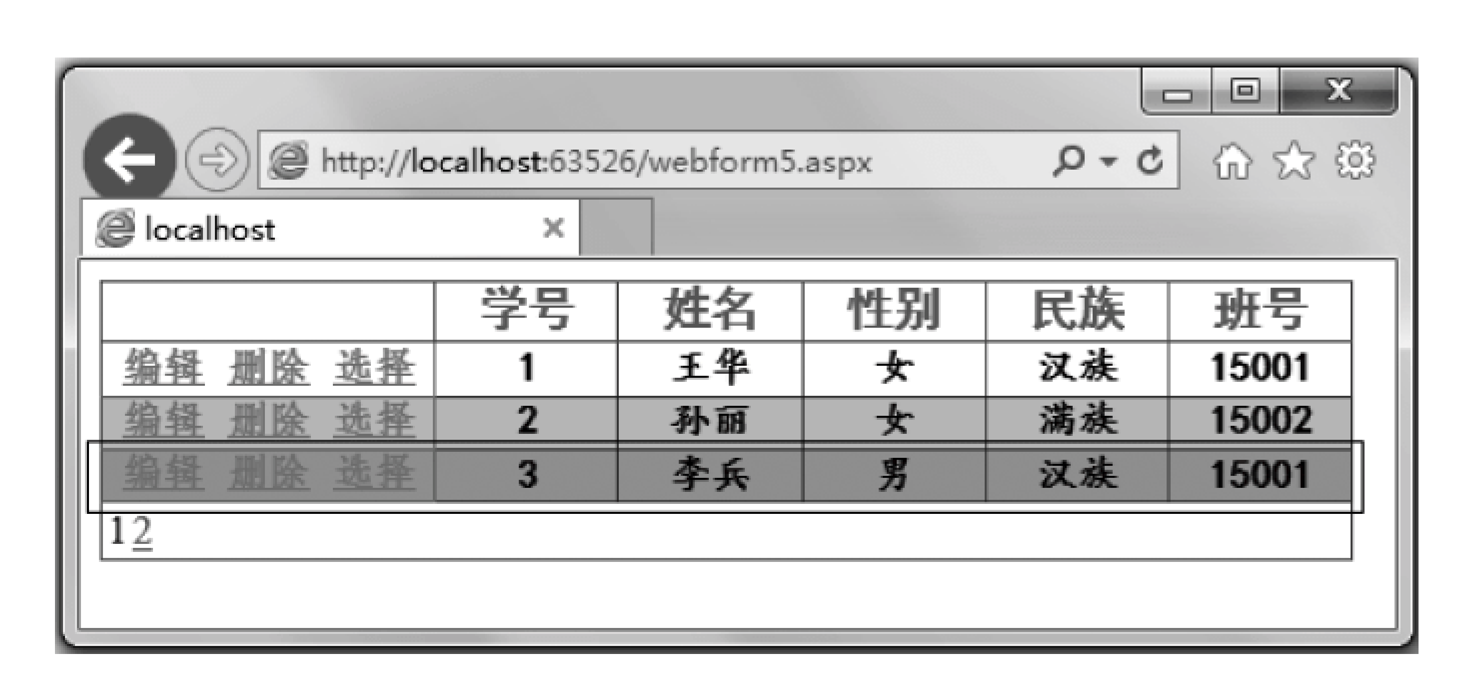


图 11.35　修改后 webform5 网页的执行界面二

2. 列模板

有时，需要的功能是系统提供的列字段类型中不具备的，需要为应用程序添加列模板。

添加列模板有两种方式：第一种是通过“字段”对话框中“可用字段”列表中选取 TemplateField 选项，单击“添加”按钮，就在“选定的字段”列表中创建了一个模板列；第二种方式是通过把一个“选定的字段”列表中现有的列转换为模板列，通过单击“将些字段转换为 TemplateField”链接完成。

在完成模板的转换后，选择“GridView 任务”中的“编辑模板”命令，就可以编辑该模板。TemplateField 类包含如表 11.14 所示的多个模板。

表 11.14　模板类型

模板类型	说　明
HeaderTemplate	在列头部显示一个单元格，即列头部模板
FootTemplate	在列尾部显示一个单元格，即列尾部模板
ItemTemplate	显示模式下的每行单元格，即列显示模板
AlternatiingItemTemplate	显示模式下的隔行单元格，即隔行模板
EditItemTemplate	编辑模式下的单元格，即编辑模板
EmptyDataTemplate	当没有相应数据时，呈现给用户的外观表示，即列空数据模板
PagerTemplate	分页模式下呈现的外观，即分页模板

例如，在 GridView 控件中单击命令域的"编辑"按钮时，进入记录编辑模式，但默认的编辑模式外观不美观，如图 11.36 所示，课程名的编辑框太长。



图 11.36　记录编辑外观

修改编辑模式外观的操作是，进入"字段"对话框，在"选定的字段"单击要修改的字段，单击"将此字段转换为 TemplateField"链接，单击"确定"按钮退出"字段"对话框。展开"GridView 任务"列表，选择"编辑模板"命令，选中该字段，显示其所有模板如图 11.37 所示，此时开发人员可以进行修改，如将 EditItemTemplate 中的文本框改短。

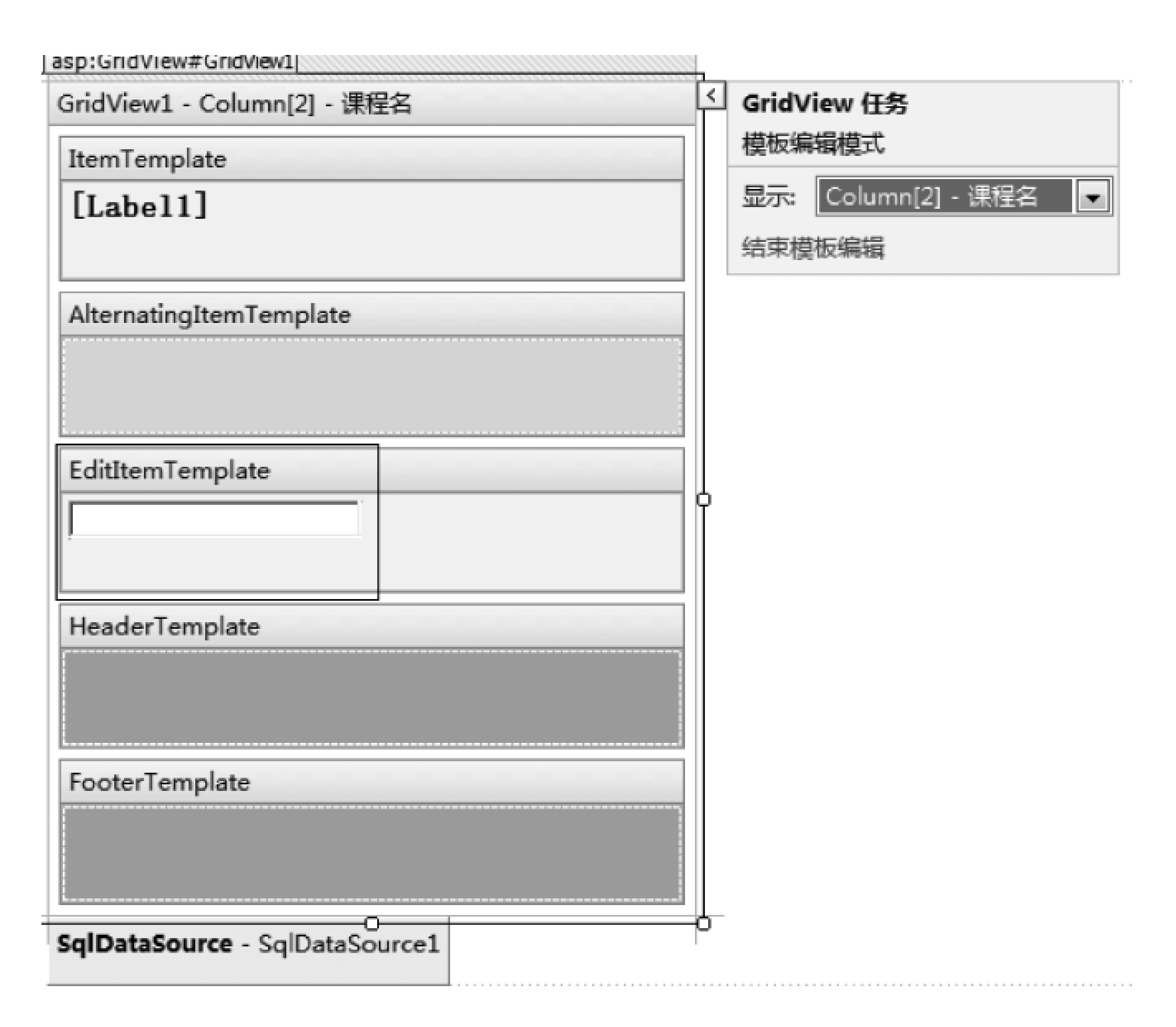


图 11.37　课程名列的 TemplateField

【练一练】　在本章 CH11 网站中添加一个 webform6 网页，其功能是类似 webform4，增加分页和分数转换为等级的功能。

其设计步骤如下：

① 打开 CH11 网站，选择"网站|添加新项"菜单命令，出现"添加新项-CH11"对话框，在中间列表中选择"Web 窗体"，将文件名称改为 webform6.aspx，其他保持默认项，单击"添加"按钮。

② 采用 webform4 网页的步骤②～⑦。

③ 设置 GridView1 控件的 AllowPaging 属性为 True，PageSize 属性为 5。

④ 从 GridView1 控件的“GridView 任务”列表中选择“编辑列”命令，出现“字段”对话框，从“可用字段”列表中选中 TemplateField，单击“添加”按钮将其加入到“选定的字段”列表中，选中它，在 TemplateField 属性中将 HeaderText 属性改为“等级”，如图 11.38 所示。

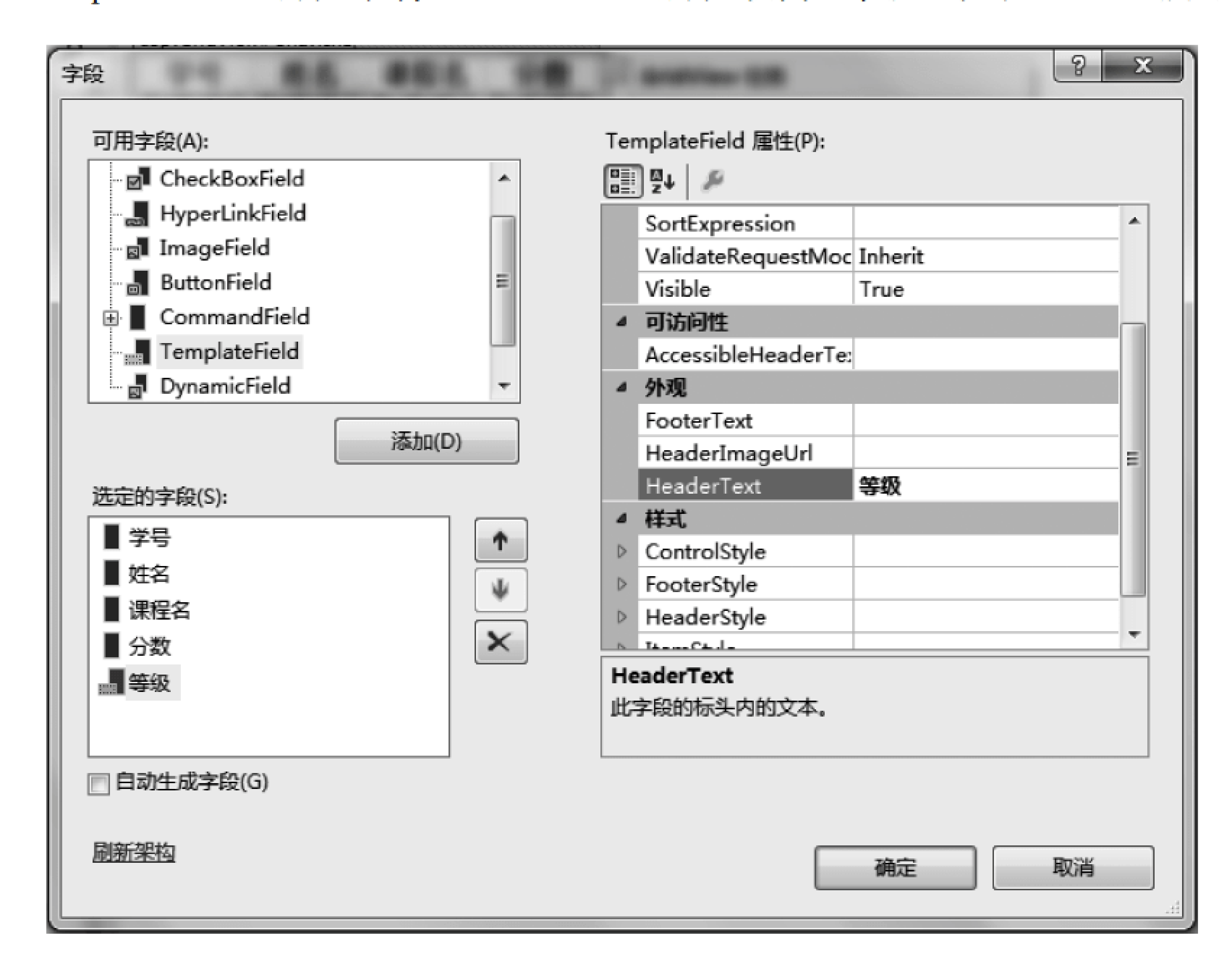


图 11.38　“字段”对话框

⑤ 从 GridView1 控件的“GridView 任务”列表中选择“编辑模板”命令，出现如图 11.39 所示的“模板编辑模式”列表，从中选择 Column[4]下面的 ItemTemplate。

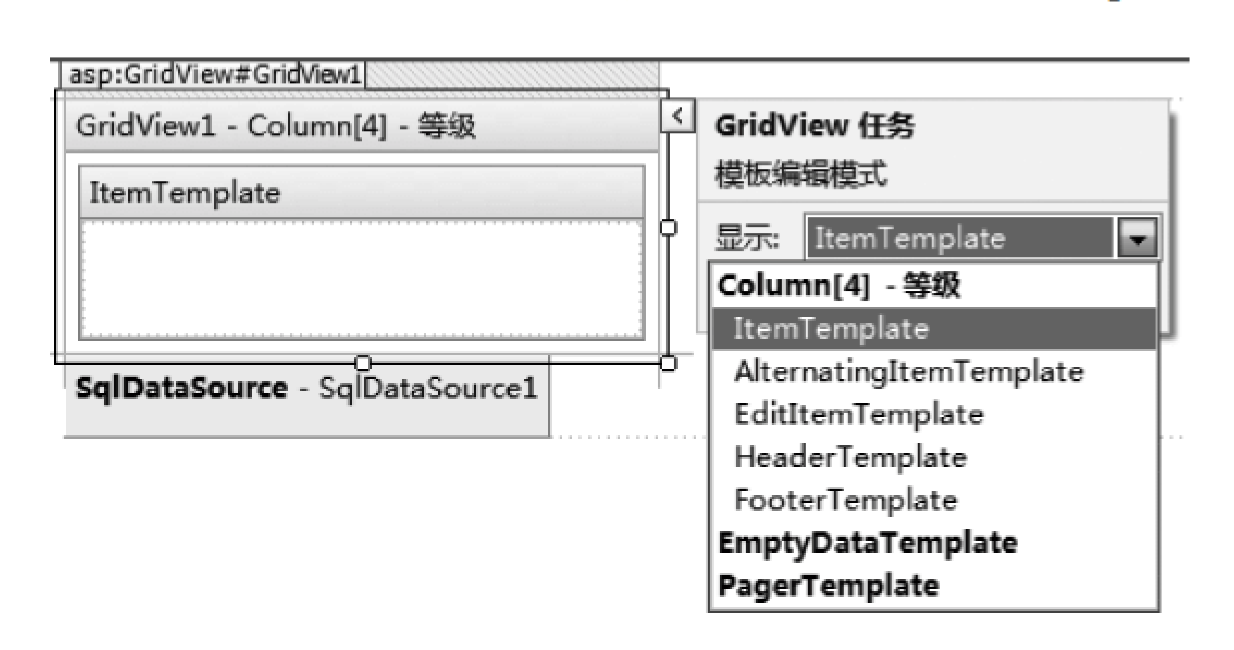


图 11.39　“模板编辑模式”列表

⑥ 在 ItemTemplate 区中拖曳一个 Label 控件，从“Label 任务”列表中选择“编辑 DataBindings”命令，打开 Label1 DataBindings 对话框，在“可绑定属性”中选择 Text，并选中“自定义绑定”选项，在代码表达式文本框中输入 GetLevel(Eval("分数"))，如图 11.40 所示。单击“确定”按钮。返回后再单击“GridView 任务”列表中的“结束编辑模板”命令。

注意： 数据绑定表达式包含在“<%”和“%>”分隔符之内，并使用 Eval 或 Bind 函数，Eval 函数用于定义单向(只读)绑定；Bind 函数用于定义双向(可更新)绑定；在调用控件或

Page 对象的 DataBind 方法时，会对数据绑定表达式进行解析。当控件仅显示值时，可以使用 Eval 方法。当用户可以修改数据值时，可以使用 Bind 方法。如果模板包含值可能被用户更改的控件（如 TextBox 或 CheckBox 控件），或模板允许删除记录，则需使用 Bind 方法。



图 11.40　Label1 DataBindings 对话框

⑦ 在本网页上设计如下事件过程：

```
public string GetLevel(object s)
{    int fs = int.Parse(s.ToString());
     if (fs >= 90)
         return("优");
     else if (fs >= 80)
         return("良");
     else if (fs >= 70)
         return("中");
     else if (fs >= 60)
         return("及格");
     else
         return("不及格");
}
```

此时看到网页中 GridView1 控件的源视图代码如下：

```
<asp:GridView ID = "GridView1" runat = "server" AutoGenerateColumns = "False"
     DataKeyNames = "学号" DataSourceID = "SqlDataSource1"
     AllowPaging = "True" PageSize = "5">
     <Columns>
        <asp:BoundField DataField = "学号" HeaderText = "学号" ReadOnly = "True"
            SortExpression = "学号" />
        <asp:BoundField DataField = "姓名" HeaderText = "姓名" SortExpression = "姓名" />
        <asp:BoundField DataField = "课程名" HeaderText = "课程名"
            SortExpression = "课程名" />
```

```
            <asp:BoundField DataField = "分数" HeaderText = "分数" SortExpression = "分数" />
            <asp:TemplateField HeaderText = "等级">
              <ItemTemplate>
                  <asp:Label ID = "Label1" runat = "server"
                       Text = '<% # GetLevel(Eval("分数")) %>'></asp:Label>
              </ItemTemplate>
            </asp:TemplateField>
        </Columns>
</asp:GridView>
```

⑧ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，其执行界面如图 11.41 所示。从中看到，等级列会自动调用 GrtLevel 方法产生对应的结果。



图 11.41　webform6 网页的执行界面

3. DataBinder 类

前例中使用了动态绑定，实际上是使用 DataBinder 类实现的，该类提供对应用程序快速开发设计器的支持以生成和分析数据绑定表达式语法。在网页数据绑定语法中可以使用此类的静态方法 Eval 在运行时计算数据绑定表达式。与标准数据绑定相比，这提供的语法更容易记忆，但是因为 DataBinder. Eval 提供自动类型转换，这会导致服务器响应时间变长。

Eval 方法的基本语法格式如下：

```
public static Object Eval(Object container,string expression)
```

其中，参数 container 指出进行计算的对象引用；expression 指出从 container 到要放置在绑定控件属性中的公共属性值的导航路径。其返回值为 Object，它是数据绑定表达式的计算结果。

必须将“<%＃”和“%>”标记放在数据绑定表达式的两头；这些标记也用于标准 ASP .NET 数据绑定。当数据绑定到模板列表中的控件时，此方法尤其有用。

对于所有的列表 Web 控件，如 DataGrid、DataList 或 Repeater，container 参数值均应为 Container. DataItem。如果要对页进行绑定，则 container 参数值应为 Page。

例如，使用 Eval 方法以绑定到 Price 字段，其使用代码如下：

```
<% # DataBinder.Eval(Container.DataItem, "Price") %>
```

4. GridViewRow 类

GridView 控件中的单独行就是一个 GridViewRow 对象。

GridView 控件将其所有数据行都存储在 Rows 集合中。若要确定 Rows 集合中 GridViewRow 对象的索引，需使用 RowIndex 属性。

通过使用 Cells 属性，可以访问 GridViewRow 对象的单独单元格。如果某个单元格包含其他控件，则通过使用单元格的 Controls 集合，可以从单元格检索控件。如果控件指定了 ID，还可以使用单元格的 FindControl 方法来查找该控件。

若要从 BoundField 字段列或自动生成的字段列检索字段值，需使用单元格的 Text 属性。

若要从将字段值绑定到控件的其他字段列类型检索字段值，先从相应的单元格检索控件，然后访问该控件的相应属性。

下面的示例演示如何使用 GridViewRow 对象，从 GridView 控件中的单元格检索字段值，然后在网页上显示该值：

```
GridViewRow selectRow = GridView1.SelectedRow;
String txt = selectRow.Cells[1].Text;
```

【练一练】 在本章 CH11 网站中添加一个 webform7 网页，其功能是修改 student 表记录的班号。

其设计步骤如下：

① 打开 CH11 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH11”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform7.aspx，其他保持默认项，单击“添加”按钮。

② 向网页中拖曳一个 GridView 控件 GridView1。在其下方拖曳一个 Button 控件 Button1。

③ 在“GridView 任务”列表中选择“自动套用格式”命令设置 GridView1 控件的外观为“石板”，并按照前面例子的方法设置其连接字符串为 SchoolConnectionString，并选中“指定自定义 SQL 语句或存储过程”复选框，进入“定义自定义语句或存储过程”页面，输入 SELECT 语句如图 11.42 所示，输入 UPDATE 语句如图 11.43 所示，单击“下一步”按钮，再单击“完成”按钮返回。这样建立了 SqlDataSource1 控件，并自动将 GridView1 控件的 DataSourceID 设置为 SqlDataSource1。

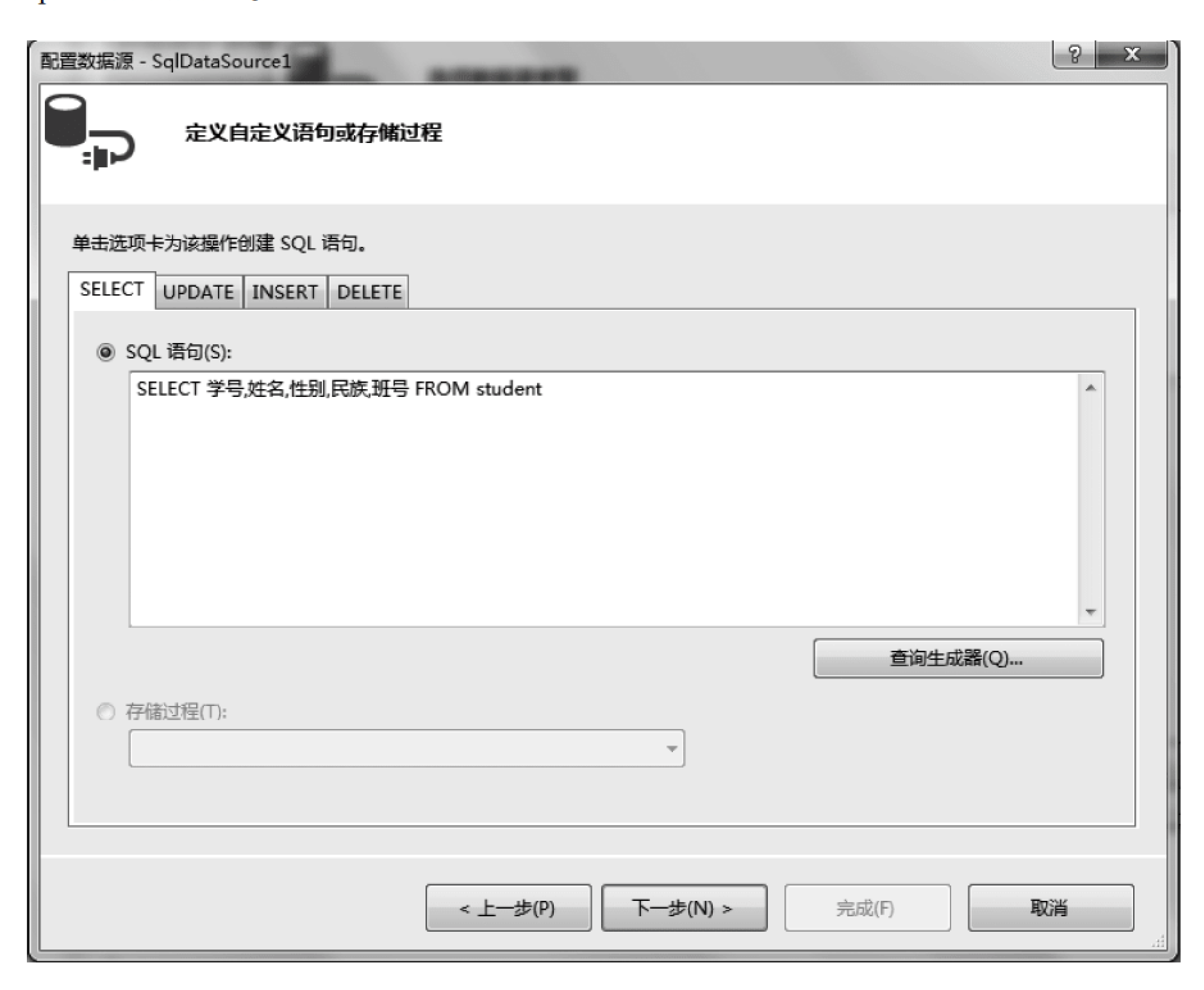


图 11.42　设置 SELECT 语句



图 11.43　设置 UPDATE 语句

此时，SqlDataSource1 控件的源视图代码如下：

```
<asp:SqlDataSource ID = "SqlDataSource1" runat = "server"
    ConnectionString = "<%$ ConnectionStrings:schoolConnectionString %>"
    SelectCommand = "SELECT 学号,姓名,性别,民族,班号 FROM student"
    UpdateCommand = "UPDATE student SET 班号 = @bh WHERE 学号 = @xh">
    <UpdateParameters>
        <asp:Parameter Name = "bh" />
        <asp:Parameter Name = "xh" />
    </UpdateParameters>
</asp:SqlDataSource>
```

④ 在“GridView 任务”列表中选择“编辑列”命令，打开“字段”对话框，从“选定的字段”列表中选中“班号”，单击“将此字段转换为 TemplateField”链接，单击“确定”按钮返回。

⑤ 在“GridView 任务”列表中选择“编辑模板”命令，指定“Columns[4]-班号”下的 ItemTemplate 模板，删除原来的 Label1 控件，向其中拖曳一个 TextBox 控件 TextBox1，如图 11.44 所示。从“TextBox 任务”列表中选择“编辑 DataBindings”命令，打开 TextBox1 DataBindings 对话框，选择“自定义绑定”单选按钮，在代码表达式文本框中输入 DataBinder.Eval(Container.DataItem,"班号")，如图 11.45 所示。单击“确定”按钮。返回后再单击“GridView 任务”列表中的“结束编辑模板”命令。

此时，GridView1 控件的源视图代码如下：

```
<asp:GridView ID = "GridView1" runat = "server" AutoGenerateColumns = "False"
    BackColor = "White" BorderColor = "#E7E7FF" BorderStyle = "None"
    BorderWidth = "1px" CellPadding = "3" DataKeyNames = "学号"
```

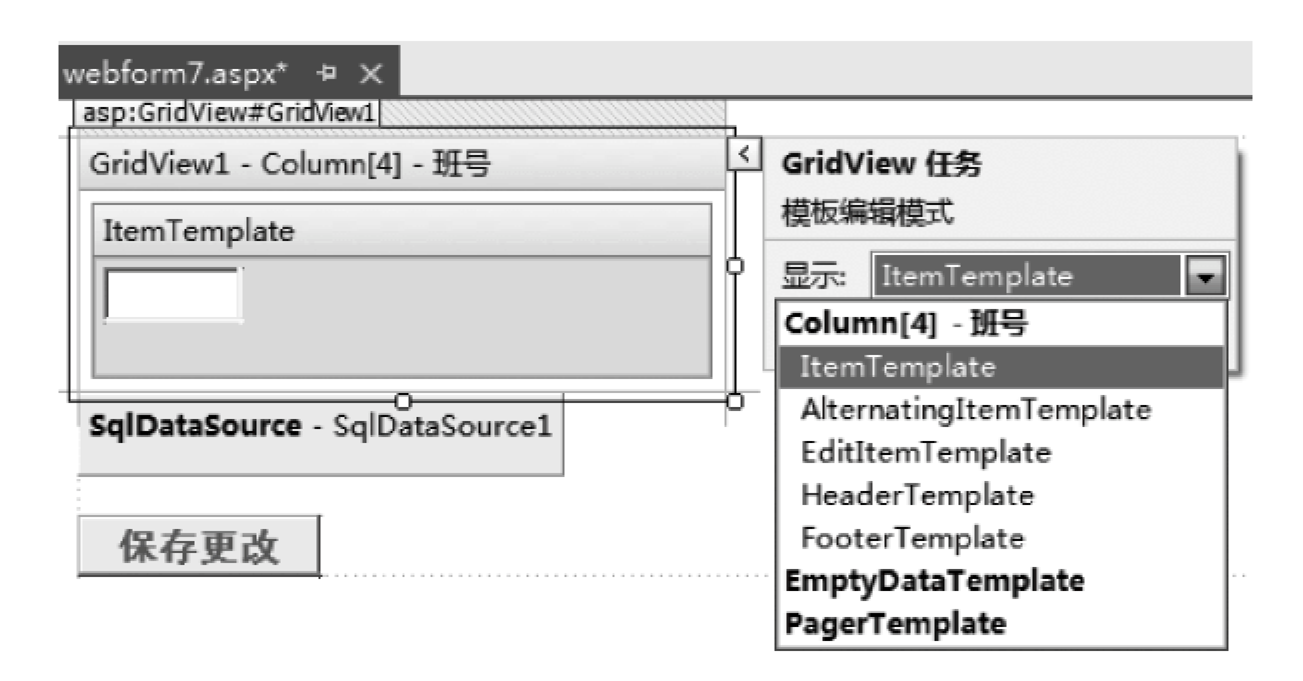


图 11.44　定义 ItemTemplate 模板



图 11.45　TextBox1 DataBindings 对话框

```
DataSourceID = "SqlDataSource1" GridLines = "Horizontal" Width = "272px">
<AlternatingRowStyle BackColor = "#F7F7F7" />
<Columns>
  <asp:BoundField DataField = "学号" HeaderText = "学号" ReadOnly = "True"
      SortExpression = "学号" />
  <asp:BoundField DataField = "姓名" HeaderText = "姓名" SortExpression = "姓名" />
  <asp:BoundField DataField = "性别" HeaderText = "性别" SortExpression = "性别" />
  <asp:BoundField DataField = "民族" HeaderText = "民族" SortExpression = "民族" />
  <asp:TemplateField HeaderText = "班号" SortExpression = "班号">
    <EditItemTemplate>
      <asp:TextBox ID = "TextBox1" runat = "server" Text = '<%# Bind("班号") %>' />
    </EditItemTemplate>
    <ItemTemplate>
      <asp:TextBox ID = "TextBox1" runat = "server" Height = "17px"
       Text = '<%# DataBinder.Eval(Container.DataItem,"班号") %>' Width = "50px" />
    </ItemTemplate>
  </asp:TemplateField>
</Columns>
```

```
        <FooterStyle BackColor="#B5C7DE" ForeColor="#4A3C8C" />
        <HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeColor="#F7F7F7" />
        <PagerStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" HorizontalAlign="Right" />
        <RowStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" />
        <SelectedRowStyle BackColor="#738A9C" Font-Bold="True" ForeColor="#F7F7F7" />
        <SortedAscendingCellStyle BackColor="#F4F4FD" />
        <SortedAscendingHeaderStyle BackColor="#5A4C9D" />
        <SortedDescendingCellStyle BackColor="#D8D8F0" />
        <SortedDescendingHeaderStyle BackColor="#3E3277" />
</asp:GridView>
```

⑥ 在本网页中设计如下事件过程：

```
protected void Button1_Click(object sender, EventArgs e)
{   string xh;
    TextBox txtbh;
    int i;
    for (i = 0; i < GridView1.Rows.Count; i++)
    {   txtbh = GridView1.Rows[i].FindControl("TextBox1") as TextBox;
        //在该行中找 TextBox1 控件
        xh = GridView1.Rows[i].Cells[0].Text;    //提取该行的学号
        Update(xh, txtbh.Text);                  //调用自定义过程进行更新
    }
}
protected void Update(string xh, string nbh)
//自定义过程,调用 SqlDataSource1 控件的 Upadte 方法
{   SqlDataSource1.UpdateParameters["xh"].DefaultValue = xh;
    SqlDataSource1.UpdateParameters["bh"].DefaultValue = nbh;
    SqlDataSource1.Update();
}
```

在 Button1_Click 事件处理方法中，GridView1. Rows[i]就是一个 GridViewRow 对象，表示 GridView1 控件索引为 i 的行对象。txtbh = GridView1. Rows[i]. FindControl("TextBox1") as TextBox 语句的功能是查找到用户选择的那行的 TextBox1 控件。

在自定义的更新方法中，通过 SqlDataSource1. UpdateParameters["xh"]. DefaultValue 设置 xh 参数的值，然后调用 SqlDataSource1 控件的 Update()方法，从而更新 student 数据表。

⑦ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，用户可以直接修改各记录的班号，如将所有班号后加 1，如图 11.46 所示，然后单击“保存更改”按钮，则数据库中所有记录发生更改。为了后面的例子，将所有数据更改恢复，如图 11.47 所示，再单击“保存更改”按钮。



图 11.46　webform7 网页的执行界面一



图 11.47　webform7 网页的执行界面二

注意：本例不像直接使用GridView控件的编辑功能，只能一次修改一个记录，而是一次可以修改多个记录，达到批量数据修改的目的，在实际Web应用程序设计中十分实用。

5. GridView控件的事件设计

GridView控件提供了一系列的事件，包含数据绑定、行绑定、行编辑、行更新、行删除、分页、鼠标和排序等事件。根据这些事件的引发时刻设计相应的事件处理方法是十分重要的。

由于数据源控件封装了许多功能，不便于理解GridView控件的事件设计原理，下面几个例子采用第10章介绍的数据库访问方法进行编程。

(1) 分页事件设计

【练一练】 在本章CH11网站中添加一个webform8网页，其功能是利用GridView控件的相关事件实现数据分页。

其设计步骤如下：

① 打开CH11网站，选择"网站|添加新项"菜单命令，出现"添加新项-CH11"对话框，在中间列表中选择"Web窗体"，将文件名称改为webform8.aspx，其他保持默认项，单击"添加"按钮。

② 向网页中拖曳一个GridView控件GridView1，在"GridView任务"列表中选择"自动套用格式"命令，设置GridView1控件的外观为"石板"。将其AllowPaging属性设置为true(允许分页)，PageSize属性设置为4。

③ 在本网页上设计如下事件处理方法：

```
using System.Data.SqlClient;
using System.Data;
protected void Page_Load(object sender, EventArgs e)
{   string mystr, mysql;
    SqlConnection myconn = new SqlConnection();
    SqlCommand mycmd = new SqlCommand();
    mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["schoolConnectionString"].ToString();
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT * FROM score";
    DataSet myds = new DataSet();
    SqlDataAdapter myda = new SqlDataAdapter(mysql,myconn);
    myda.Fill(myds, "score");
    myconn.Close();
    GridView1.DataSource = myds.Tables["score"];
    GridView1.DataBind();
}
```

④ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，初始界面如图11.48所示，但单击页号2时出现错误，如图11.49所示。

如果指定GridView1控件的DataSourceID为SqlDataSource1控件时，由于SqlDataSource1控件具有分页功能，所以这样做不会有问题。本例采用的是DataSet对象，它不具备分页功能。

解决的方法是设计相应的GridView1_PageIndexChanging事件处理方法，在用户分页操作时重新绑定。修改本网页的事件处理方法如下：

```
protected void Page_Load(object sender, EventArgs e)
{    if (!Page.IsPostBack)
     {
         bind();                                   //在网页首发时执行
     }
}
protected void bind()                              //自定义绑定方法
{    string mystr, mysql;
     SqlConnection myconn = new SqlConnection();
     SqlCommand mycmd = new SqlCommand();
     mystr = System.Configuration.ConfigurationManager.
         ConnectionStrings["schoolConnectionString"].ToString();
     myconn.ConnectionString = mystr;
     myconn.Open();
     mysql = "SELECT * FROM score";
     DataSet myds = new DataSet();
     SqlDataAdapter myda = new SqlDataAdapter(mysql, myconn);
     myda.Fill(myds, "score");
     myconn.Close();
     GridView1.DataSource = myds.Tables["score"];
     GridView1.DataBind();
}
protected void GridView1_PageIndexChanging(object sender,
       System.Web.UI.WebControls.GridViewPageEventArgs e)
{    //分页事件处理
     GridView1.PageIndex = e.NewPageIndex;
     bind();                                       //在分页时调用自定义方法
}
```

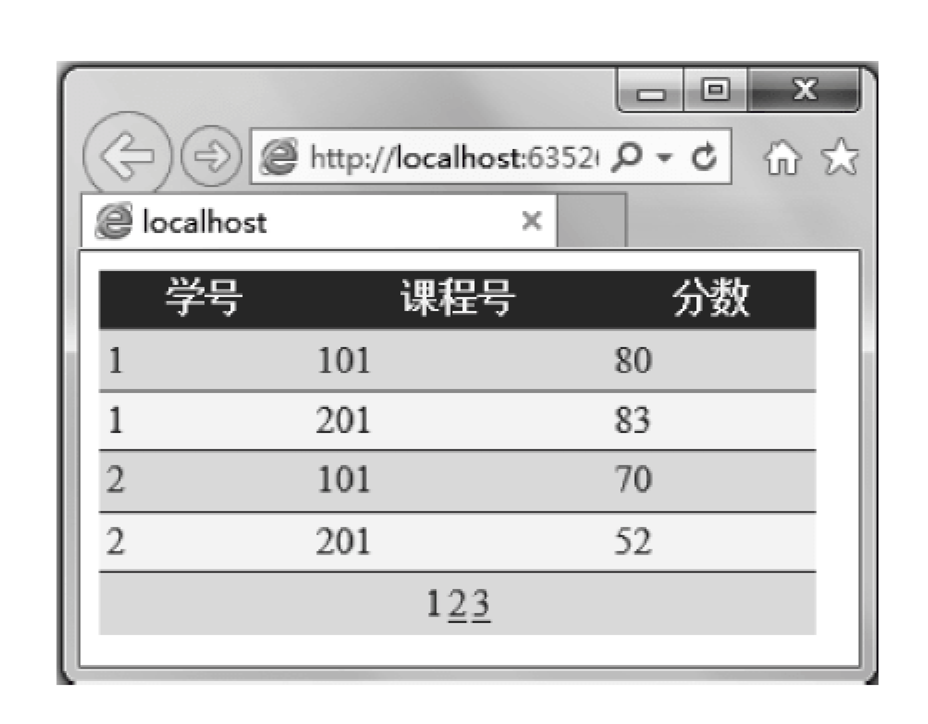


图 11.48　webform8 网页的执行界面一



图 11.49　webform8 网页的执行界面二

在 GridView1_PageIndexChanging 事件处理方法中，参数 e 为 GridViewPageEventArgs 类对象，其 NewPageIndex 属性用于获取或设置要在 GridView 控件中显示的新页的索引。该事件处理方法在用户单击某一页导航按钮时，在 GridView 控件处理分页操作之前执行，将 GridView1 控件的当前页索引 PageIndex 设置为新页的索引。这样在用户分页操作时，将当前页号改为 e.NewPageIndex，从而能够正确实现分页。

(2) 行绑定事件设计

【练一练】　在本章 CH11 网站中添加一个 webform9 网页，它在 webform8 网页的基础上增加功能：分数低于 60，用红色显示；分数高于 90(含)，用绿色显示；其他用黄色显示。

其设计步骤如下：

① 打开 CH11 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH11”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform9. aspx，其他保持默认项，单击“添加”按钮。

② 本网页的设计与 webform8 的设计相同。

③ 本网页包含 webform8 的所有事件处理方法，另外增加如下事件处理方法：

```
protected void GridView1_RowDataBound(object sender,
     System.Web.UI.WebControls.GridViewRowEventArgs e)
{    if (e.Row.RowIndex >= 0)
     {    float fs = float.Parse(e.Row.Cells[2].Text.Trim());              //提取该行的分数
          if (fs < 60)
               e.Row.BackColor = System.Drawing.Color.Red;
          else if (fs < 90)
               e.Row.BackColor = System.Drawing.Color.Yellow;
          else
               e.Row.BackColor = System.Drawing.Color.BlueViolet;
     }
}
```

在呈现 GridView 控件之前，该控件中的每一行必须绑定到数据源中的一条记录。将某个数据行绑定到 GridView 控件中的数据以后，将引发 RowDataBound 事件。开发人员可以提供一个这样的事件处理方法，即每次发生此事件时就执行一个自定义例程(如修改绑定到该行的数据的值)。其中参数 e 为 GridViewRowEventArgs 类对象，表示引发事件的那个 GridView 控件的行(GridViewRow 对象)。可以使用该参数访问正在绑定行的属性，若要访问行中的特定单元格，可以使用参数 e 的 Row 属性，该属性中包含的 GridViewRow 对象的 Cells 属性。还可以通过使用 GridViewRow 对象的 RowType 属性，确定要行类型(是标头行还是数据行等)。

④ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，如图 11.50 所示，看到不同分数的显示效果。



图 11.50　webform9 网页的执行界面

(3) 行操作事件设计

【练一练】　在本章 CH11 网站中设计两个网页 webform10 和 webform10-1，用于说明行编辑事件的设计方法。

其设计步骤如下：

① 打开 CH11 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH11”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform10. aspx，其他保持默认项，单击“添加”按钮。

② 在 webform10 网页中放入一个 HTML 文字和一个 GridView1 控件。展开 GridView1 控件的“GridView 任务”列表，指定其自动套用格式为“沙滩和天空”。单击“编辑列”进入“字段”对话框，从“可用字段”中添加 3 个 BoundField 字段到“选定字段”中，将第 1 个字段的 DataField 和 HeaderText 属性均设置为“学号”，第 2 个字段的 DataField 和 HeaderText 属性均设置为“课程号”，第 3 个字段的 DataField 和 HeaderText 属性均设置为“分数”。再从“可用字段”中添加 3 个 CommandField 字段到“选定字段”中，其属性设置如

图 11.51 所示,只保留"编辑"按钮。将这 4 个字段的 ItemStyle 下的 HorizontalAlign 属性均设置为 Center。

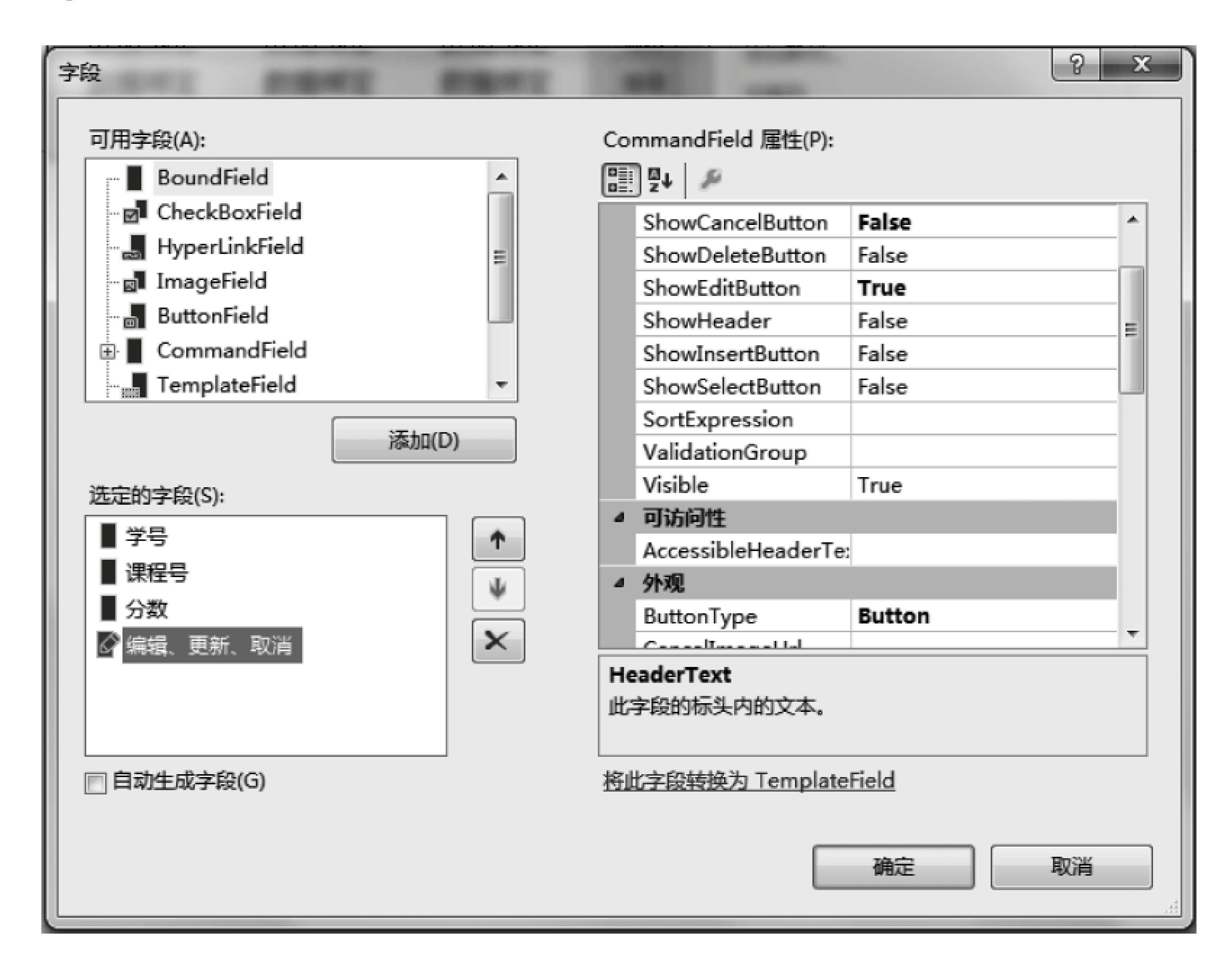


图 11.51　设置 CommandField 字段的属性

③ 将 GridView1 控件的 AllowPaging 属性设置为 True,PageSize 属性设置为 5,AutoGenerateColumns 属性设置为 False。

④ 本网页包含 webform8 的所有事件处理方法,另外增加如下事件处理方法:

```
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{   string xh = GridView1.Rows[e.NewEditIndex].Cells[0].Text;
    Session["xh"] = xh;
    string kch = GridView1.Rows[e.NewEditIndex].Cells[1].Text;
    Session["kch"] = kch;
    string fs = GridView1.Rows[e.NewEditIndex].Cells[2].Text;
    Session["fs"] = fs;
    Response.Redirect("webform10 - 1.aspx");      //转向 webform10 - 1 网页
}
```

在用户单击某一行的"编辑"按钮后,在 GridView 控件进入编辑模式之前,将引发 RowEditing 事件。开发人员可以提供一个事件处理方法,即每次发生此事件时就执行一个自定义例程(如取消编辑操作或调用自己的编辑网页)。其中,参数 e(GridViewEditEventArgs 类对象)将传递给事件处理方法,其 NewEditIndex 属性获取或设置所编辑行的索引。若要取消编辑操作,可以将参数 e 的 Cancel 属性设置为 True。

⑤ 再添加一个名称为 webform10-1 的网页。其设计界面如图 11.52 所示,主要包含 3 个文本框和 2 个命令按钮(Button1 和 Button2)。文本框从上到下分别为 TextBox1～TextBox3,前两个为只读的。



图 11.52　webform10-1 网页的设计界面

⑥ webform10-1 网页包含的事件处理方法如下：

```
protected void Page_Load(object sender, EventArgs e)
{   if (!Page.IsPostBack)
    {   TextBox1.Text = Session["xh"].ToString().Trim();
        TextBox2.Text = Session["kch"].ToString().Trim();
        TextBox3.Text = Session["fs"].ToString().Trim();
    }
}
protected void Button1_Click(object sender, EventArgs e)
{   string mystr, mysql;
    SqlConnection myconn = new SqlConnection();
    SqlCommand mycmd = new SqlCommand();
    mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["schoolConnectionString"].ToString();
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "UPDATE score SET 分数 = " + TextBox3.Text.Trim()
          + " WHERE 学号 = " + TextBox1.Text.Trim()
          + " AND 课程号 = '" + TextBox2.Text.Trim() + "'";
    mycmd.CommandText = mysql;
    mycmd.Connection = myconn;
    mycmd.ExecuteNonQuery();
    myconn.Close();
    Response.Redirect("webform10.aspx");          //返回 webform10 网页
}
protected void Button2_Click(object sender, EventArgs e)
{
    Response.Redirect("webform10.aspx");          //返回 webform10 网页
}
```

⑦ 单击工具栏中的 ▶ Internet Explorer 按钮执行 webform10 网页，其初始界面如图 11.53 所示，用户可以进行分页操作。单击第 2 行的“编辑”按钮，出现如图 11.54 所示的界面，用户可以修改分数（不能修改学号和课程号），单击“更新”按钮会更新数据库，然后返回；单击“取消”不会更新数据库，直接返回。



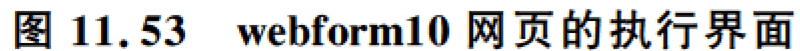

图 11.53　webform10 网页的执行界面



图 11.54　webform10-1 网页的执行界面

另外常用的还有 RowDeleting 事件。用户单击 GridView 控件中的某一行的“删除”按钮后，在 GridView 控件删除该行之前，将引发 RowDeleting 事件。开发人员可以提供一个如下

的事件处理方法，即每次发生此事件时就执行一个自定义例程（如取消删除操作）。

```
void GridView1_RowDeleting(Object sender, GridViewDeleteEventArgs e)
```

其中，参数 e(GridViewDeleteEventArgs 类对象)可以确定当前行的索引，它的 RowIndex 属性用于获取所删除行的索引。若要取消删除操作，可以将 e 的 Cancel 属性设置为 True。

6. DataView 类

在 GridView 控件实现数据绑定中经常用 DataView 对象。DataView 对象能够创建 DataTable 中所存储数据的不同视图，用于对 DataSet 中的数据进行排序、过滤和查询等操作。DataView 类对象最常用的使用方式如图 11.55 所示。

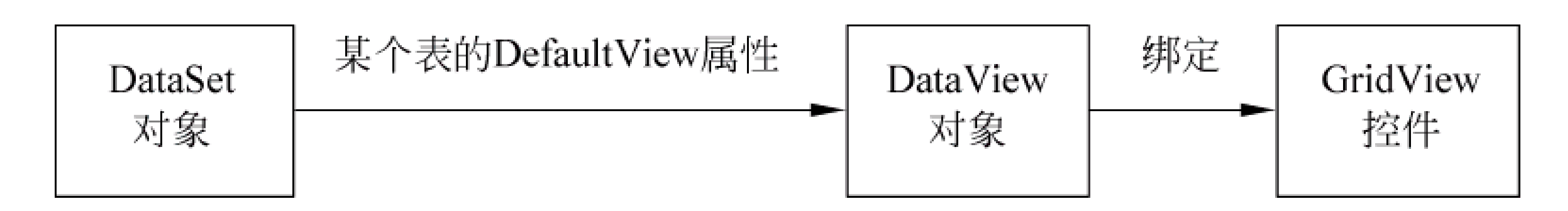


图 11.55　DataView 类对象的使用方式

DataView 对象类似于数据库中的 View 功能，提供 DataTable 列(Column)排序、过滤记录(Row)及记录的搜索，它的一个常见用法是为控件提供数据绑定。

DataView 对象的构造函数如下：

```
DataView()
DataView(table)
DataView(table, RowFilter, Sort, RowState)
```

其中，table 参数指出要添加到 DataView 的 DataTable；RowFilter 参数指出要应用于 DataView 的 RowFilter。Sort 参数指出要应用于 DataView 的 Sort；RowState 参数指出要应用于 DataView 的 DataViewRowState。

为给定的 DataTable 创建一个新的 DataView，可以声明该 DataView，把 DataTable 的一个引用 mydt 传给 DataView 构造函数。例如：

```
DataView mydv = new DataView(mydt);
```

用过滤条件属性可以得到 DataView 中数据行的一个子集合，也可以为这些数据排序。

DataTable 对象提供 DefaultView 属性返回默认的 DataView 对象。例如：

```
DataView mydv = new DataView();
mydv = myds.Tables["student"].DefaultView;
```

上述代码从 myds 数据集中取得 student 表的默认内容，再利用相关控件（如 DataGridView）显示内容，指定数据来源为 mydv。

DataView 对象的常用属性如表 11.15 所示。其常用方法如表 11.16 所示。

例如，设计一个 A 网页，获取用户输入的一个查询条件字符串 condstr，在该网页中通过如下代码调用 B 网页：

```
Server.Transfer("B.aspx?" + "condstr = " + condstr);
```

在 B 网页中设计如下事件处理方法，用于在一个 GridView1 控件中显示从 student 表查询的满足 condstr 条件的结果：

```
DataView mydv = new DataView();
protected void Page_Load(object sender, EventArgs e)
{   string condstr = Request.QueryString["condstr"];
    string mystr = ConfigurationManager.
        ConnectionStrings["schoolConnectionString"].ToString();
    SqlConnection myconn = new SqlConnection();
    myconn.ConnectionString = mystr;
    myconn.Open();
    DataSet myds = new DataSet();
    SqlDataAdapter myda = new SqlDataAdapter("SELECT * FROM student", myconn);
    myda.Fill(myds, "student");
    mydv = myds.Tables["student"].DefaultView;   //获得 DataView 对象 mydv
    mydv.RowFilter = condstr;                    //过滤 DataView 中的记录
    GridView1.DataSource = mydv;
    GridView1.DataBind();
}
```

表 11.15　DataView 的常用属性及其说明

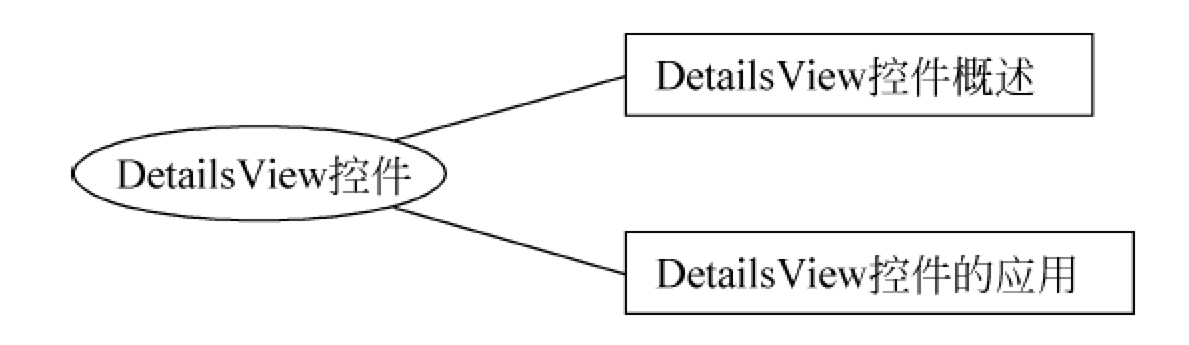
属　　性	说　　明
AllowDelete	设置或获取一个值，该值指示是否允许删除
AllowEdit	获取或设置一个值，该值指示是否允许编辑
Allownew	获取或设置一个值，该值指示是否可以使用 Addnew 方法添加新行
ApplyDefaultSort	获取或设置一个值，该值指示是否使用默认排序
Count	在应用 RowFilter 和 RowStateFilter 之后，获取 DataView 中记录的数量
Item	从指定的表获取一行数据
RowFilter	获取或设置用于筛选在 DataView 中查看哪些行的表达式
RowStateFilter	获取或设置用于 DataView 中的行状态筛选器
Sort	获取或设置 DataView 的一个或多个排序列以及排序顺序
Table	获取或设置源 DataTable

表 11.16　DataView 的常用方法及其说明

方　　法	说　　明
Addnew	将新行添加到 DataView 中
Delete	删除指定索引位置的行
Find	按指定的排序关键字值在 DataView 中查找行
FindRows	返回 DataRowView 对象的数组，这些对象的列与指定的排序关键字值匹配
ToTable	根据现有 DataView 中的行，创建并返回一个新的 DataTable

11.5　DetailsView 控件

知识梳理



11.5.1　DetailsView 控件概述

DetailsView 控件显示来自数据源单条记录的值，其中每个数据行表示该记录的一个字段。它可与 GridView 控件结合使用，用于主/详细信息方案。DetailsView 控件支持的功能有绑定到数据源控件（如 SqlDataSource），内置插入功能，内置更新和删除功能，内置分页功能，以编程方式访问 DetailsView 对象模型以动态设置属性、处理事件等。

DetailsView 控件的许多用法与 GridView 控件类似。DetailsView 控件的常用属性及其说明如表 11.17 所示，常用方法其说明如表 11.18 所示，常用事件其说明如表 11.19 所示。

表 11.17　DetailsView 控件的常用属性及其说明

属　性	说　明
AllowPaging	获取或设置一个值，该值指示是否启用分页功能
AutoGenerateDeleteButton	获取或设置一个值，该值指示用来删除当前记录的内置控件是否在 DetailsView 控件中显示
AutoGenerateEditButton	获取或设置一个值，该值指示用来编辑当前记录的内置控件是否在 DetailsView 控件中显示
AutoGenerateInsertButton	获取或设置一个值，该值指示用来插入新记录的内置控件是否在 DetailsView 控件中显示
AutoGenerateRows	获取或设置一个值，该值指示对应于数据源中每个字段的行字段是否自动生成并在 DetailsView 控件中显示
DataItem	获取绑定到 DetailsView 控件的数据项
DataItemCount	获取基础数据源中的项数
DataItemIndex	从基础数据源中获取 DetailsView 控件中正在显示的项的索引
DataKey	获取一个 DataKey 对象，该对象表示所显示的记录的主键
DataMember	当数据源包含多个不同的数据项列表时，获取或设置数据绑定控件绑定到的数据列表的名称
DataSource	获取或设置对象，数据绑定控件从该对象中检索其数据项列表
DataSourceID	获取或设置控件的 ID，数据绑定控件从该控件中检索其数据项列表
DefaultMode	获取或设置 DetailsView 控件的默认数据输入模式
GridLines	获取或设置 DetailsView 控件的网格线样式
HorizontalAlign	获取或设置 DetailsView 控件在页面上的水平对齐方式
InsertRowStyle	获取一个对 TableItemStyle 对象的引用，该对象允许设置在 DetailsView 控件处于插入模式时 DetailsView 控件中数据行的外观
PageCount	获取数据源中的记录数
PageIndex	获取或设置所显示的记录的索引
SelectedValue	获取 DetailsView 控件中的当前记录的数据键值

表 11.18　DetailsView 控件的常用方法及其说明

方　法	说　明
ChangeMode	将 DetailsView 控件切换为指定模式 newMode。其中 newMode 的取值为 DetailsViewMode. Edit（DetailsView 控件处于编辑模式，这样用户就可以更新记录的值）、DetailsViewMode. Insert（DetailsView 控件处于插入模式，这样用户就可以向数据源中添加新记录）或 DetailsView. ReadOnly（DetailsView 控件处于只读模式，这是通常的显示模式）
DataBind	将来自数据源的数据绑定到控件
DeleteItem	从数据源中删除当前记录
InsertItem	将当前记录插入到数据源中
UpdateItem	更新数据源中的当前记录

表 11.19　DetailsView 控件的常用事件及其说明

事　　件	说　　明
ItemCommand	当单击 DetailsView 控件中的按钮时引发
ItemCreated	在 DetailsView 控件中创建记录时引发
ItemDeleted	在单击 DetailsView 控件中的“删除”按钮时，但在删除操作之后引发
ItemDeleting	在单击 DetailsView 控件中的“删除”按钮时，但在删除操作之前引发
ItemInserted	在单击 DetailsView 控件中的“插入”按钮时，但在插入操作之后引发
ItemInserting	在单击 DetailsView 控件中的“插入”按钮时，但在插入操作之前引发
ItemUpdated	在单击 DetailsView 控件中的“更新”按钮时，但在更新操作之后引发
ItemUpdating	在单击 DetailsView 控件中的“更新”按钮时，但在更新操作之前引发
PageIndexChanged	当 PageIndex 属性的值在分页操作后更改时引发
PageIndexChanging	当 PageIndex 属性的值在分页操作前更改时引发

11.5.2　DetailsView 控件的应用

DetailsView 控件可以用于记录编辑，但更多的情况是与 GridView 控件关联使用，构成主细表。

【练一练】 在本章 CH11 网站中添加一个 webform11 网页，其功能是通过 GridView 控件和 DetailsView 控件来操作 student 表记录。

其设计步骤如下：

① 打开 CH11 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH11”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform11.aspx，其他保持默认项，单击“添加”按钮。

② 在本网页中插入一个 2 行 2 列的表格。第一行放置两个起提示作用的 HTML 文字。

③ 向表格的第 2 行第 1 列中拖曳一个 GridView 控件 GridView1，指定其自动套用格式为“沙滩和天空”。采用前面介绍的操作建立其数据源控件为 SqlDataSource1 控件(使用“高级”按钮生成 INSERT、UPDATE 和 DELETE 语句)，它用于连接 school 数据库的 student 表，单击“完成”按钮。通过“GridView 任务”列表选中“启动分页”、“启动排序”、“启动删除”和“启动选定内容”复选框。设置其 PageSize 属性为 4。

④ 向表格的第 2 行第 2 列中拖曳一个 DetailsView 控件 DetailsView1。指定其 DataSourceID 为 AccessDataSource1，通过“DetailsView 任务”列表选中“启动插入”和“启动编辑”复选框。指定自动套用格式为“红糖”。本网页设计界面如图 11.56 所示。



图 11.56　webform11 网页的设计界面

⑤ 在该网页上设计如下事件过程：

```
protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
{   DetailsView1.ChangeMode(DetailsViewMode.ReadOnly);
    DetailsView1.PageIndex = GridView1.PageIndex * GridView1.PageSize
        + GridView1.SelectedIndex;
}
protected void GridView1_PageIndexChanged(object sender, EventArgs e)
{   DetailsView1.ChangeMode(DetailsViewMode.ReadOnly);
    DetailsView1.PageIndex = GridView1.PageIndex * GridView1.PageSize;
}
```

由于 SqlDataSource1 控件具有选择和分页功能，在上面的方法中，使用 GridVeiw1 控件的 PageIndex 和 PageSize 来决定 DetailsView1 控件当前的页面索引。使两个控件同步。

⑥ 单击工具栏中的 ▶ Internet Explorer 按钮执行本网页，在 GridView1 控件中单击学号 3 记录的“选择”连接，DetailsView1 控件中显示该记录，如图 11.57 所示，用户可以通过“编辑”连接来修改该记录，如图 11.58 所示。

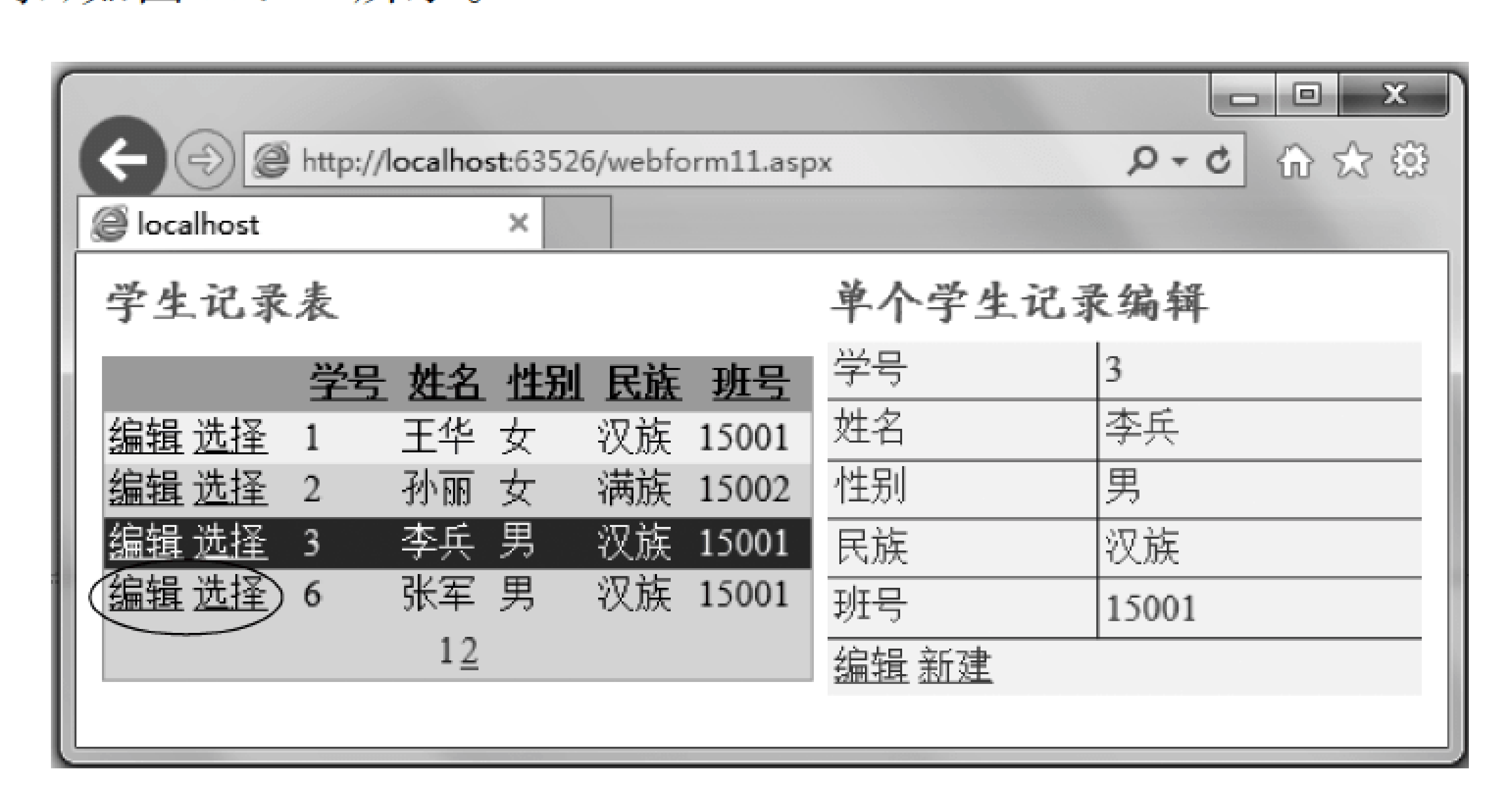


图 11.57　webform11 网页的执行界面一



图 11.58　webform11 网页的执行界面二

其他常用的数据绑定控件有 FromView 和 DataList。

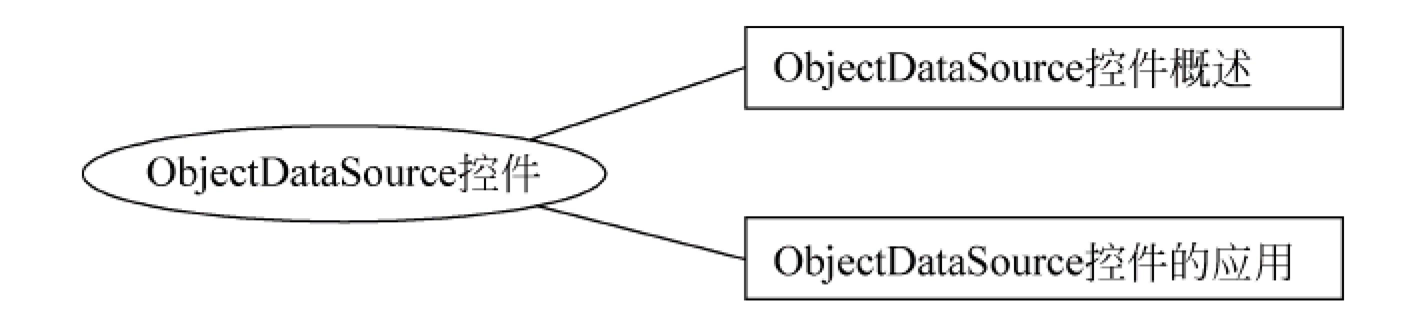
FromView 控件与 DetailsView 控件类似，可用于处理数据源中的单条记录。FormView 控件和 DetailsView 控件之间的差别在于 DetailsView 控件使用表格布局，在该布局中，记录的每个字段都各自显示为一行。FormView 控件不指定用于显示记录的预定义布局，开发人

员需要创建一个包含控件的模板，以显示记录中的各个域。该模板中包含用于创建表单的格式、控件和绑定表达式。

DataList 控件用可自定义的格式显示各行数据库信息，在所创建的模板中定义数据显示布局，可以为项、交替项、选定项和编辑项创建模板，也可以使用标题、脚注和分隔符模板自定义 DataList 控件的整体外观。通过在模板中包括 Button 命令按钮，可将列表项连接到代码，而这些代码允许用户在显示、选择和编辑模式之间进行切换。

11.6 ObjectDataSource 控件

知识梳理



11.6.1 ObjectDataSource 控件概述

ObjectDataSource 控件包含的属性等与 SqlDataSource 控件的类似，但增加了若干用于个性化的属性。

从属性上看，SqlDataSource 控件和 ObjectDataSource 控件的区别如表 11.20 所示，即 SqlDataSource 控件只能设置所执行的 SQL 语句，ObjectDataSource 控件可以设置所执行的自定义方法，而自定义方法具有更好的灵活性。

表 11.20 SqlDataSource 控件和 ObjectDataSource 控件的属性区别

SqlDataSource 的属性	ObjectDataSource 的属性
SelectCommand	SelectMethod
InsertCommand	InsertMethod
UpdateCommand	UpdateMethod
DeleteCommand	DeleteMethod

若要从网页对象中检索数据，需要用检索数据的方法的名称来设置 SelectMethod 属性，该方法通常返回一个 DataSet 对象。如果方法签名带参数，可以将 Parameter 对象添加到 SelectParameters 集合中，然后将它们绑定到要传递给由 SelectMethod 方法指定方法的值。

为使 ObjectDataSource 能够使用参数，这些参数必须与方法签名中的参数名称和类型相匹配。每次调用 Select 方法时，ObjectDataSource 控件都检索数据。此方法提供对 SelectMethod 属性所指定的方法的编程访问。当调用绑定到 ObjectDataSource 控件的 DataBind 方法时，这些控件自动调用 SelectMethod 属性指定的方法。如果设置数据绑定控件的 DataSourceID 属性，该控件根据需要自动绑定到数据源中的数据。建议通过设置 DataSourceID 属性将 ObjectDataSource 控件绑定到数据绑定控件；或可以设置 DataSource 属性，但之后必须显式调用数据绑定控件的 DataBind 方法。可以随时以编程方式调用 Select 方法来检索数据。

根据 ObjectDataSource 控件使用的网页对象的功能，可以执行数据操作，如更新、插入和删除。若要执行这些数据操作，需为要执行的操作设置适当的方法名称和任何关联的参数。例如，对于更新操作，将 UpdateMethod 属性设置为业务对象方法的名称，该方法执行更新并将所需的任何参数添加到 UpdateParameters 集合中。如果 ObjectDataSource 控件与数据绑定控件相关联，则由数据绑定控件添加参数。这种情况下，需要确保方法的参数名称与数据绑定控件中的字段名称相匹配。调用更新 Update 方法时，由代码显式执行更新或由数据绑定控件自动执行更新。删除和插入操作遵循相同的常规模式。假定业务对象以逐个记录（而不是以批处理）的方式执行这些类型的数据操作。

由 SelectMethod、UpdateMethod、InsertMethod 和 DeleteMethod 属性标识的方法可以是实例方法或 static 方法。如果方法为 static，则不创建网页对象的实例，也不引发 ObjectCreating、ObjectCreated 和 ObjectDisposing 事件。

如果数据集作为 DataSet、DataView 或 DataTable 对象返回，ObjectDataSource 控件可以筛选由 SelectMethod 属性检索的数据。可以使用格式字符串语法将 FilterExpression 属性设置为筛选表达式，并将表达式中的值绑定到 FilterParameters 集合中指定的参数。

11.6.2　ObjectDataSource 控件的应用

ObjectDataSource 控件所执行的数据库操作方法是自定义的。在实现数据库操作的方法中，所涉及的数据表可能是单个表也可能是一组相关表。

【练一练】　在本章 CH11 网站中添加一个 webform12 网页，其功能是其功能是在 GridView 控件中显示指定班号的学生记录，并可以选择、更新和删除记录，当删除一个学生记录时需同时删除该生的所有成绩记录。

其设计步骤如下：

① 打开 CH11 网站，选择“网站|添加新项”菜单命令，出现“添加新项-CH11”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 webform12. aspx，其他保持默认项，单击“添加”按钮。

② 单击“网站|添加新项”菜单命令，选中“类”选项，新建类文件 Class1. cs（放在 App_Code 目录中），其中包含 StudentDB 类的代码如下：

```
using System.Data;
using System.Data.SqlClient;
public class StudentDB
{   public StudentDB()                          //构造函数
    { }
    public DataSet ExecuteQuery(string sql)
    {   string mystr = System.Configuration.ConfigurationManager.
            ConnectionStrings["schoolConnectionString"].ToString();
        SqlConnection myconn = new SqlConnection();
        myconn.ConnectionString = mystr;
        myconn.Open();
        SqlDataAdapter myda = new SqlDataAdapter(sql, myconn);
        DataSet myds = new DataSet();
        myda.Fill(myds);
        myconn.Close();
        return myds;
    }
    public DataSet SelectData(string bh)
```

```
    {   string mystr = System.Configuration.ConfigurationManager.
            ConnectionStrings["schoolConnectionString"].ToString();
        using (SqlConnection myconn = new SqlConnection())
        {   myconn.ConnectionString = mystr;
            myconn.Open();
            SqlCommand mycmd = new SqlCommand();
            mycmd.CommandText = "SELECT 学号,姓名,性别,民族,班号 " +
                "FROM student WHERE 班号 = @sbh ";
            mycmd.Connection = myconn;
            mycmd.Parameters.Add("@sbh", SqlDbType.VarChar, 5).Value = bh;
            using (SqlDataAdapter myda = new SqlDataAdapter())
            {   myda.SelectCommand = mycmd;
                DataSet myds = new DataSet();
                myda.Fill(myds, "student");      //将 student 表填充到 myds 中
                return myds;
            }
        }
    }
    public void UpdateData(int 学号, string 姓名, string 性别,
        string 民族, string 班号)
    {   string mystr, mysql;
        mystr = System.Configuration.ConfigurationManager.
            ConnectionStrings["schoolConnectionString"].ToString();
        mysql = "UPDATE student SET 姓名 = @姓名,性别 = @性别,民族 = @民族," +
            "班号 = @班号 WHERE 学号 = @学号";
        using (SqlConnection myconn = new SqlConnection(mystr))
        using (SqlCommand mycmd = new SqlCommand(mysql, myconn))
        {   mycmd.Parameters.Add("@姓名", SqlDbType.VarChar, 10).Value = 姓名;
            mycmd.Parameters.Add("@性别", SqlDbType.VarChar, 2).Value = 性别;
            mycmd.Parameters.Add("@民族", SqlDbType.VarChar, 10).Value = 民族;
            mycmd.Parameters.Add("@班号", SqlDbType.VarChar, 6).Value = 班号;
            mycmd.Parameters.Add("@学号", SqlDbType.Int, 5).Value = 学号;
            myconn.Open();
            mycmd.ExecuteNonQuery();
        }
    }
    public void DeleteData(int 学号)
    {   string mystr, mysql;
        mystr = System.Configuration.ConfigurationManager.
            ConnectionStrings["schoolConnectionString"].ToString();
        mysql = "DELETE FROM student WHERE 学号 = @学号";
        using (SqlConnection myconn = new SqlConnection(mystr))
        using (SqlCommand mycmd = new SqlCommand(mysql, myconn))
        {   mycmd.Parameters.Add("@学号", SqlDbType.Int, 5).Value = 学号;
            myconn.Open();
            mycmd.ExecuteNonQuery();
            mysql = "DELETE FROM score WHERE 学号 = @学号";
            using (SqlCommand mycmd1 = new SqlCommand(mysql, myconn))
            {   mycmd1.Parameters.Add("@学号",SqlDbType.VarChar,5).Value = 学号;
                mycmd1.ExecuteNonQuery();
            }
        }
    }
}
```

上述代码中,StudentDB 类的 SelectData 的方法返回一个 DataSet 对象,其中存储指定班号的所有学生记录。using 语句用于定义一个范围,它的功能是在范围末尾释放定义的对象,

这里使用 using 语句定义一个 myconn 对象，在其范围外自动释放它，所以代码中没有包含 myconn. Close()语句。

StudentDB 类的 UpdateData 方法用于更新学生记录，DeleteData 方法用于删除学生记录，它们中都使用了 using 语句，含义与 SelectData 中的相同。

③ 在 webform12 网页中放置两个 HTML 标签，一个下拉列表 DropDownList1、一个命令按钮 Button1 和一个 GridView1 控件，再将一个 ObjectDataSource 控件 ObjectDataSource1 拖曳到本网页中。

④ 展开 ObjectDataSource1 控件的“ObjectDataSource 任务”列表，单击“配置数据源”启动“配置数据源”向导。首先出现“选择业务对象”页面，从“选择业务对象”下拉列表中选择 StudentDB，如图 11.59 所示，单击“下一步”按钮。

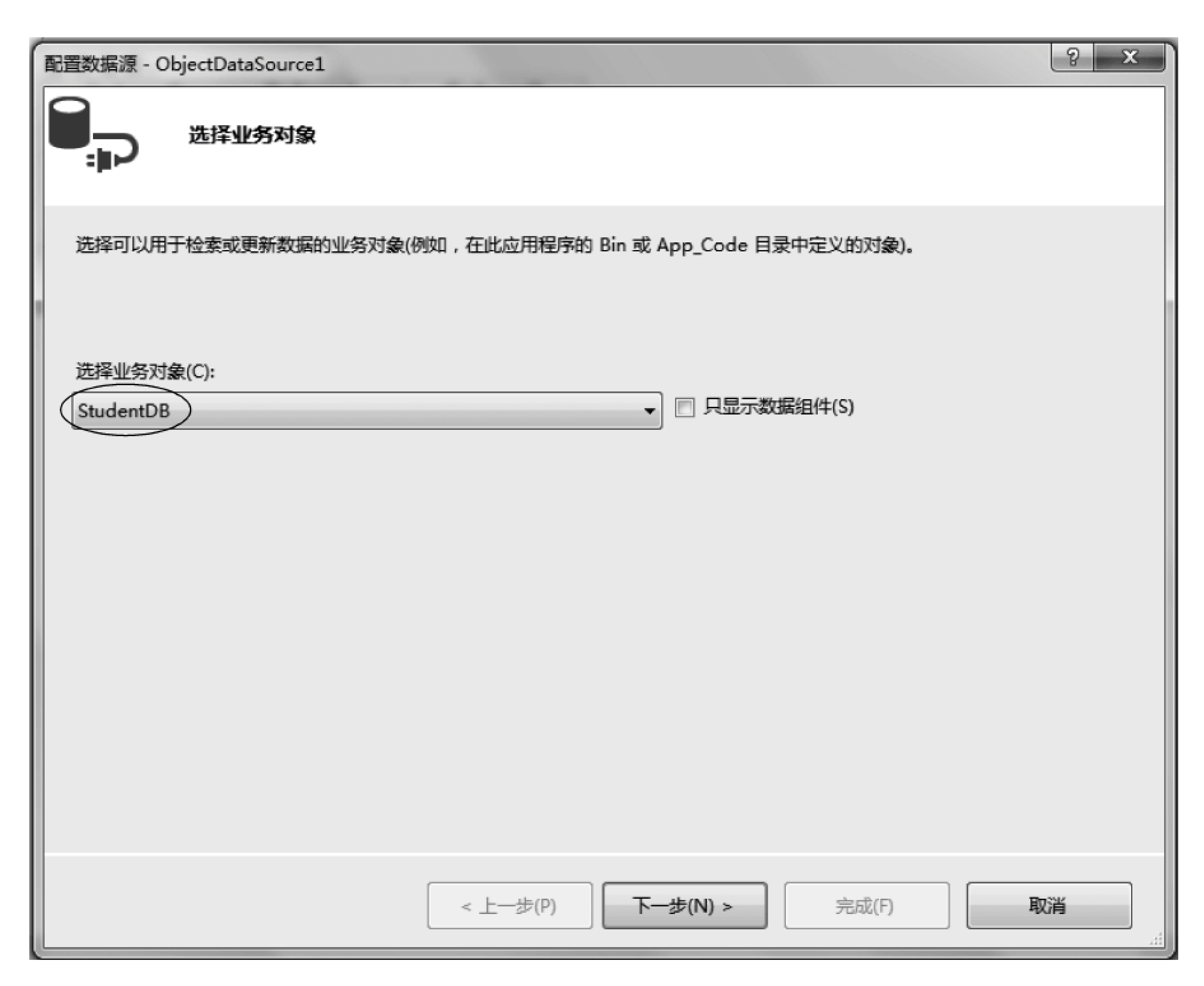


图 11.59　“选择业务对象”页面

⑤ 出现“定义数据方法”对话框，默认为 SELECT 选项卡，指定与 SELECT 操作关联并返回数据业务对象的方法为 StudentDB 类的 SelectData 方法，如图 11.60 所示。

单击 UPDATE 选项卡，指定与 UPDATE 操作关联并返回数据业务对象的方法为 StudentDB 类的 UpdateData 方法，如图 11.61 所示。

单击 DELETE 选项卡，指定与 DELETE 操作关联并返回数据业务对象的方法为 StudentDB 类的 DeleteData 方法，如图 11.62 所示。单击“下一步”按钮。

⑥ 出现“定义参数”页面，为 SelectData 方法定义参数，该方法只有一个参数 bh，它的值来自 DropDownList1 下拉列表中用户选择的文本值，所以在“参数”列表中选中它，在“参数源”下拉列表中选中 Control，在 ControlID 下拉列表中选中 DropDownList1，在 DefaultValue 文本框中输入默认值 15001，如图 11.63 所示。单击“完成”按钮。



图 11.60 SELECT 选项卡



图 11.61 UPDATE 选项卡

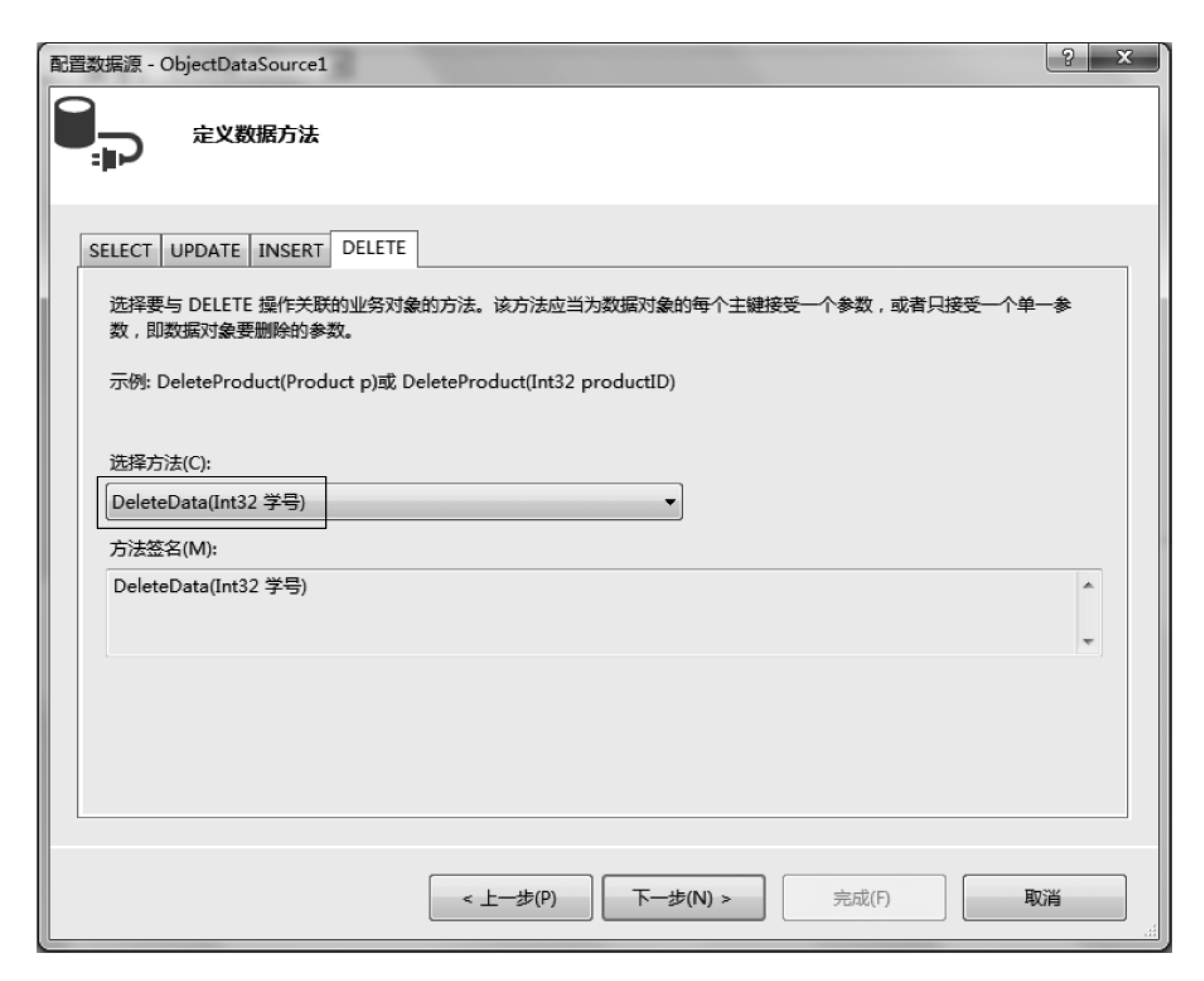


图 11.62　DELETE 选项卡

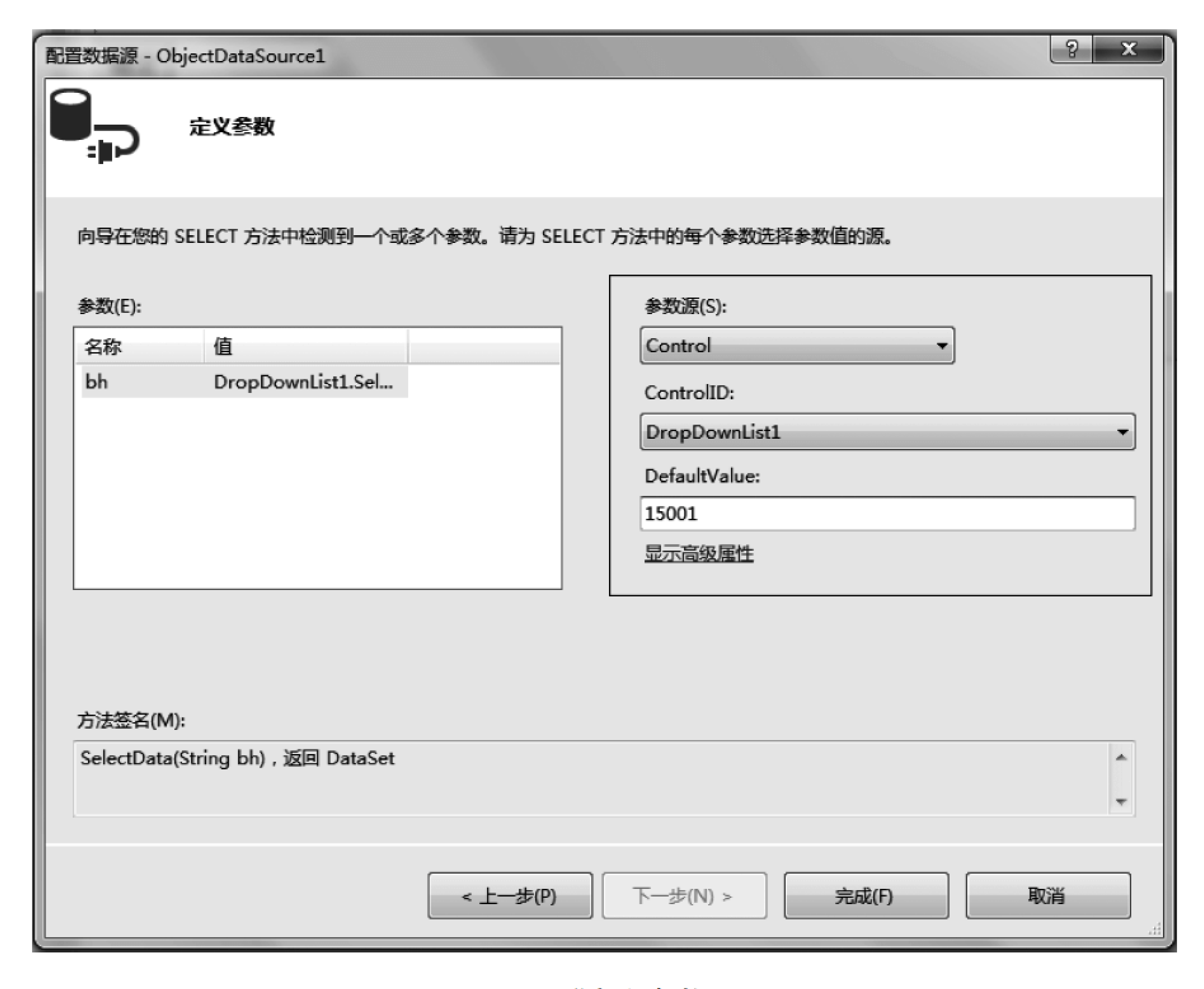


图 11.63　“定义参数”页面

说明：这里只能为 SelectData 方法指定参数，其他方法的参数在源视图代码中自动生成。

此时 ObjectDataSource1 控件的源视图代码如下：

```
<asp:ObjectDataSource ID = "ObjectDataSource1" runat = "server"
    DeleteMethod = "DeleteData"
    SelectMethod = "SelectData" TypeName = "StudentDB"
    UpdateMethod = "UpdateData">
    <DeleteParameters>
       <asp:Parameter Name = "学号" Type = "Int32" />
    </DeleteParameters>
    <SelectParameters>
       <asp:ControlParameter ControlID = "DropDownList1" DefaultValue = "15008"
          Name = "bh" PropertyName = "SelectedValue" Type = "String" />
    </SelectParameters>
    <UpdateParameters>
       <asp:Parameter Name = "学号" Type = "Int32" />
       <asp:Parameter Name = "姓名" Type = "String" />
       <asp:Parameter Name = "性别" Type = "String" />
       <asp:Parameter Name = "民族" Type = "String" />
       <asp:Parameter Name = "班号" Type = "String" />
    </UpdateParameters>
</asp:ObjectDataSource>
```

⑦ 展开 GridView1 控件的"GridView 任务"列表，单击"编辑列"命令，在"选定的字段"列表中添加 5 个 BoundField 字段和一个 CommandField 字段。

将第 1 个 BoundField 字段的 HeaderText 属性和 DataField 属性均设置为"学号"，ReadOnly 属性置为 True(学号字段不能修改)。

将第 2 个 BoundField 字段的 HeaderText 属性和 DataField 属性均设置为"姓名"。

将第 3 个 BoundField 字段的 HeaderText 属性和 DataField 属性均设置为"性别"。

将第 4 个 BoundField 字段的 HeaderText 属性和 DataField 属性均设置为"民族"。

将第 5 个 BoundField 字段的 HeaderText 属性和 DataField 属性均设置为"班号"。

将这 5 个字段都通过单击"将此字段转换为 TemplateField"链接转换为相应的 TemplateField 字段，在编辑模板中修改各个字段的 EditItemTemplate 中的文本框宽度(学号字段是只读的，所以为标签)。

将 CommandField 字段的 ButtonType 属性置为 Button。

⑧ 将 GridView1 控件的 DataSourceID 属性设置为 ObjectDataSource1，在出现的对话框中选择"否"。并"GridView 任务"列表中选中"启动分页"、"启动编辑"、"启动删除"和"启动选定内容"复选框。将其 PageSize 置为 2，DataNameKeys 属性为"学号"，将其自动套用样式置为"沙滩和天空"，并设置各个字段的 HorizontalAlign 属性为 Center，以及相应的字体和颜色。将 GridView1 控件的 DataKeyNames 属性设置为"学号"。

此时 GridView1 控件的源视图代码如下：

```
<asp:GridView ID = "GridView1" runat = "server" Width = "496px" AllowPaging = "True"
    AutoGenerateColumns = "False" BackColor = "LightGoldenrodYellow" BorderColor = "Tan"
    BorderWidth = "1px" CellPadding = "2" DataSourceID = "ObjectDataSource1"
    ForeColor = "Black" GridLines = "None" PageSize = "2" DataKeyNames = "学号">
    <AlternatingRowStyle BackColor = "PaleGoldenrod" />
    <Columns>
      <asp:TemplateField HeaderText = "学号">
```

```
        <EditItemTemplate>
          <asp:Label ID="Label1" runat="server" Height="16px"
            Text='<%# Eval("学号") %>'></asp:Label>
        </EditItemTemplate>
        <ItemTemplate>
          <asp:Label ID="Label1" runat="server" Height="16px"
            Text='<%# Bind("学号") %>'></asp:Label>
        </ItemTemplate>
        <HeaderStyle Font-Names="黑体" HorizontalAlign="Center" />
        <ItemStyle HorizontalAlign="Center" Font-Bold="True" Font-Names="楷体" />
    </asp:TemplateField>
    <asp:TemplateField HeaderText="姓名">
        <EditItemTemplate>
            <asp:TextBox ID="TextBox1" runat="server" Height="16px"
              Text='<%# Bind("姓名") %>' Width="62px"></asp:TextBox>
        </EditItemTemplate>
        <ItemTemplate>
            <asp:Label ID="Label2" runat="server" Height="16px"
              Text='<%# Bind("姓名") %>'></asp:Label>
        </ItemTemplate>
        <HeaderStyle Font-Names="黑体" HorizontalAlign="Center" />
        <ItemStyle HorizontalAlign="Center" Font-Bold="True" Font-Names="楷体" />
    </asp:TemplateField>
    <asp:TemplateField HeaderText="性别">
        <EditItemTemplate>
            <asp:TextBox ID="TextBox2" runat="server" Height="16px"
              Text='<%# Bind("性别") %>' Width="48px"></asp:TextBox>
        </EditItemTemplate>
        <ItemTemplate>
            <asp:Label ID="Label3" runat="server" Height="16px"
               Text='<%# Bind("性别") %>'></asp:Label>
        </ItemTemplate>
        <HeaderStyle Font-Bold="True" Font-Names="黑体" HorizontalAlign="Center"/>
        <ItemStyle HorizontalAlign="Center" Font-Bold="True" Font-Names="楷体" />
    </asp:TemplateField>
    <asp:TemplateField HeaderText="民族">
        <EditItemTemplate>
          <asp:TextBox ID="TextBox3" runat="server" Height="16px"
            Text='<%# Bind("民族") %>' Width="62px"></asp:TextBox>
        </EditItemTemplate>
        <ItemTemplate>
          <asp:Label ID="Label4" runat="server" Height="16px"
            Text='<%# Bind("民族") %>'></asp:Label>
        </ItemTemplate>
        <HeaderStyle Font-Names="黑体" HorizontalAlign="Center" />
        <ItemStyle HorizontalAlign="Center" Font-Bold="True" Font-Names="楷体" />
    </asp:TemplateField>
    <asp:TemplateField HeaderText="班号">
        <EditItemTemplate>
            <asp:TextBox ID="TextBox4" runat="server" Height="16px"
               Text='<%# Bind("班号") %>' Width="63px"></asp:TextBox>
        </EditItemTemplate>
        <ItemTemplate>
            <asp:Label ID="Label5" runat="server" Height="16px"
               Text='<%# Bind("班号") %>'></asp:Label>
        </ItemTemplate>
        <HeaderStyle Font-Names="黑体" HorizontalAlign="Center" />
        <ItemStyle HorizontalAlign="Center" Font-Bold="True" Font-Names="楷体" />
```

```
            </asp:TemplateField>
            <asp:CommandField ButtonType="Button" ShowDeleteButton="True"
                ShowEditButton="True" ShowSelectButton="True">
                <ControlStyle Font-Bold="True" ForeColor="Red" />
                <ItemStyle HorizontalAlign="Center" />
            </asp:CommandField>
        </Columns>
        <FooterStyle BackColor="Tan" />
        <HeaderStyle BackColor="Tan" Font-Bold="True" />
        <PagerStyle BackColor="PaleGoldenrod" ForeColor="DarkSlateBlue"
            HorizontalAlign="Center" />
        <SelectedRowStyle BackColor="DarkSlateBlue" ForeColor="GhostWhite" />
        <SortedAscendingCellStyle BackColor="#FAFAE7" />
        <SortedAscendingHeaderStyle BackColor="#DAC09E" />
        <SortedDescendingCellStyle BackColor="#E1DB9C" />
        <SortedDescendingHeaderStyle BackColor="#C2A47B" />
    </asp:GridView>
```

⑨ 为了方便测试,进入 SQL Server,在 student 表中插入 3 个 15008 班的学生记录,在 score 表中插入 2 个 122 学号学生的成绩记录。单击工具栏中的 ▶ Internet Explorer 按钮执行本网页,其初始界面如图 11.64 所示。选择班号为 15008,单击"确定"命令按钮,执行界面如图 11.65 所示。



图 11.64　webform12 网页的执行界面一

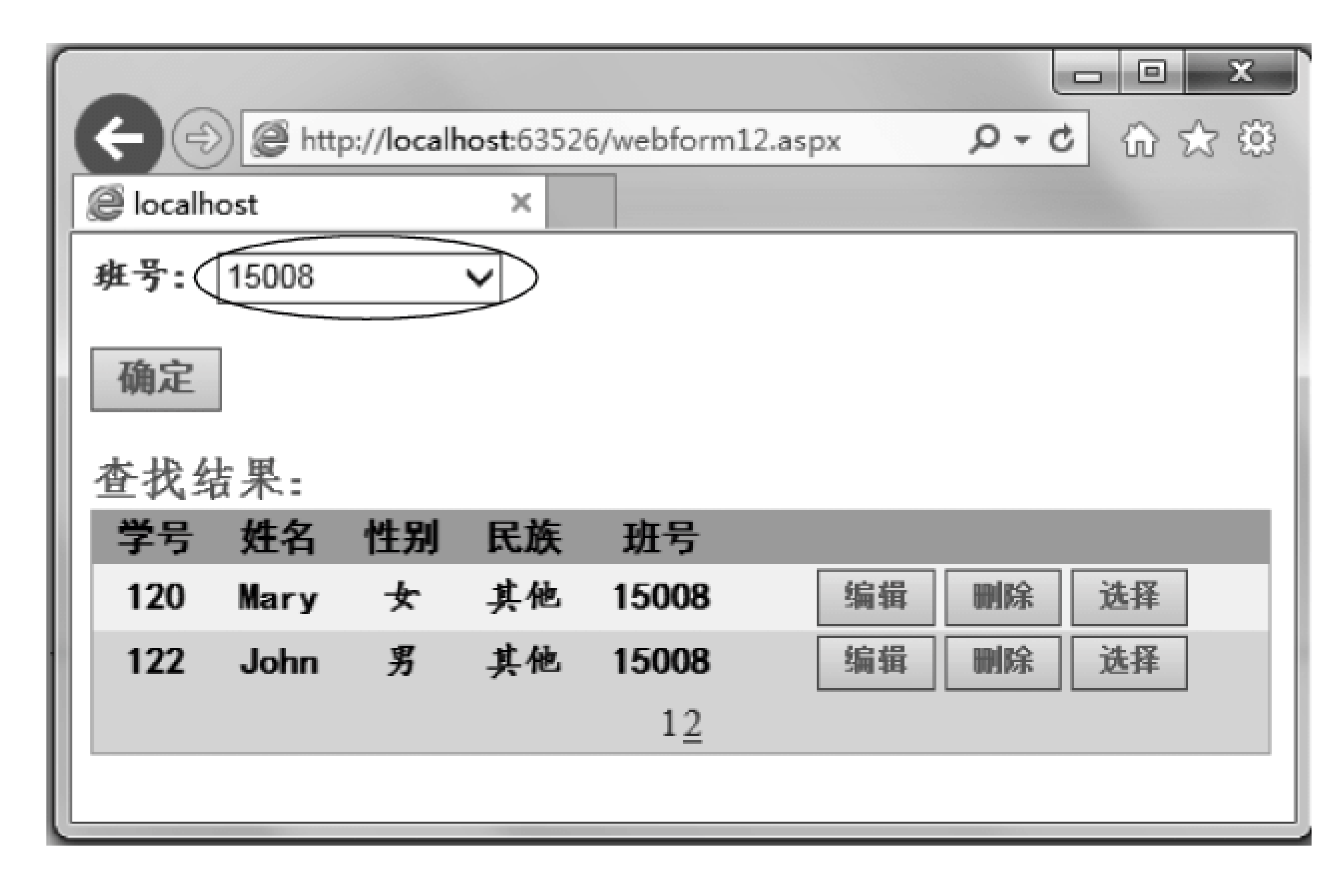


图 11.65　webform12 网页的执行界面二

单击学号 120 所在行的“编辑”按钮，出现如图 11.66 所示的界面，用户可以更新该行的数据(除了学号外)。例如，将姓名改为 Mary1，单击“更新”按钮，出现如图 11.67 所示的界面，表明记录得到更新。

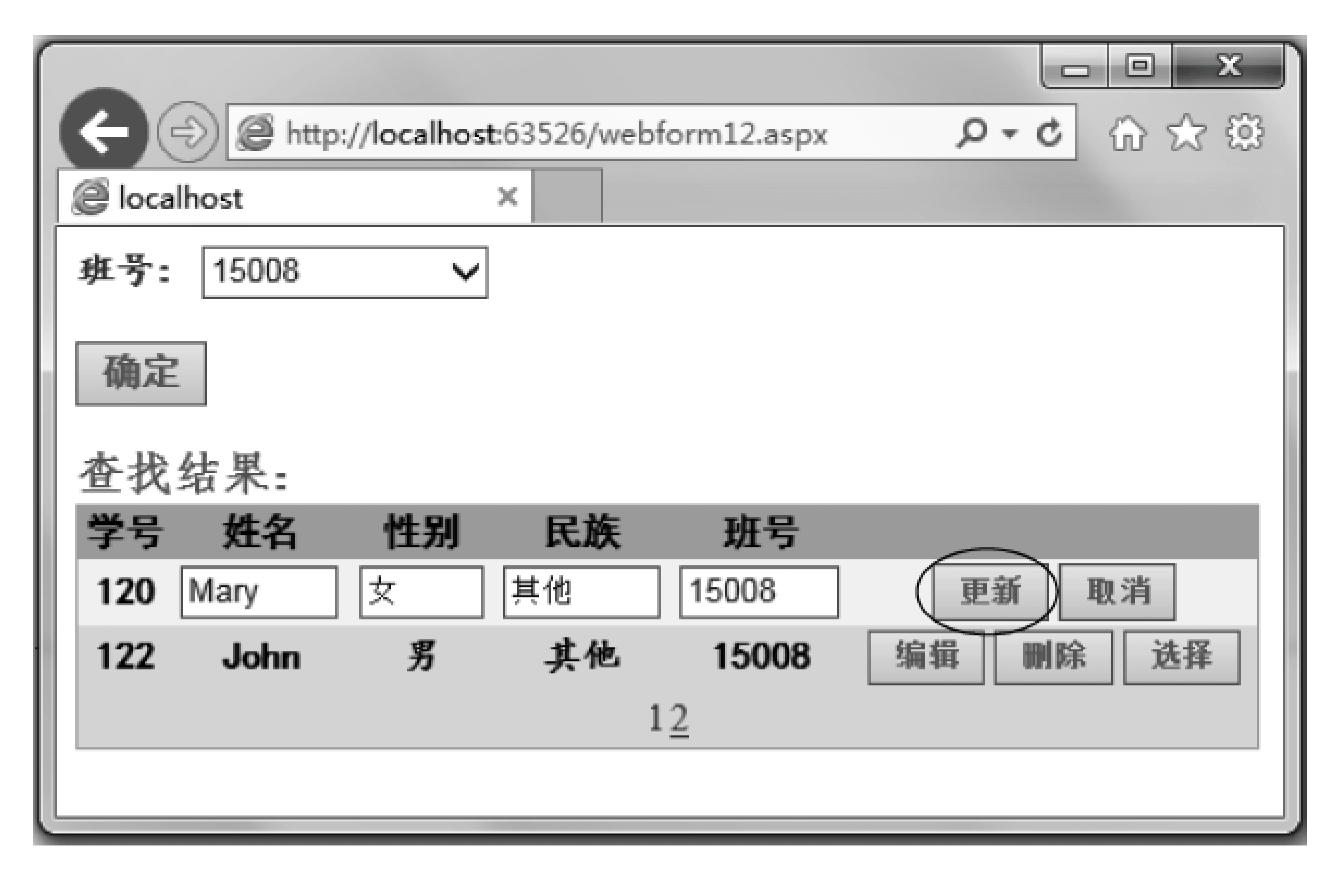


图 11.66　webform12 网页的执行界面三

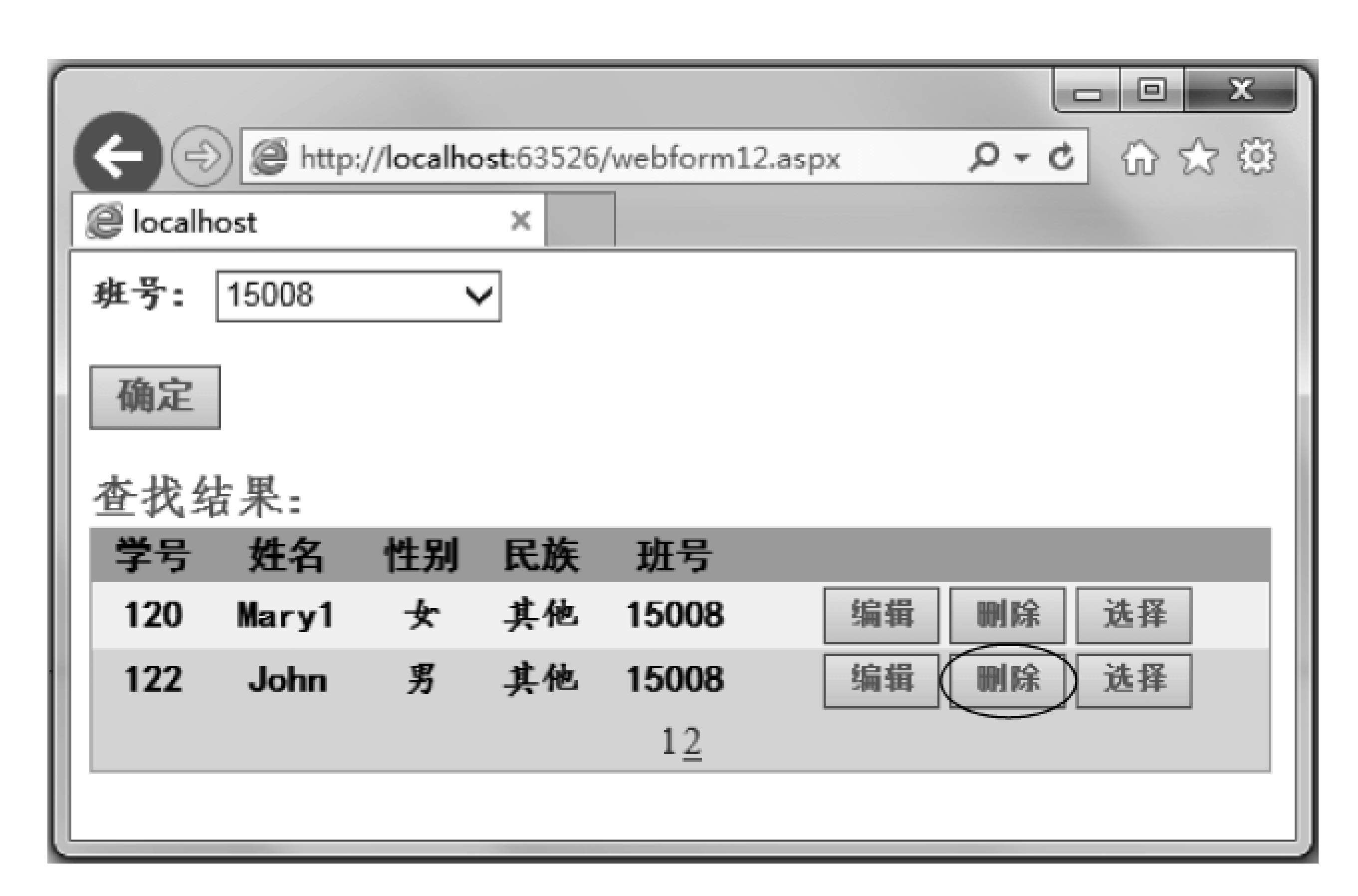


图 11.67　webform12 网页的执行界面四

再单击学号 122 所在行的“删除”按钮，出现如图 11.68 所示的界面，表明该记录被删除了。进入 SQL Server，会看到 score 表中 2 个 122 学号学生的成绩记录也被删除了。

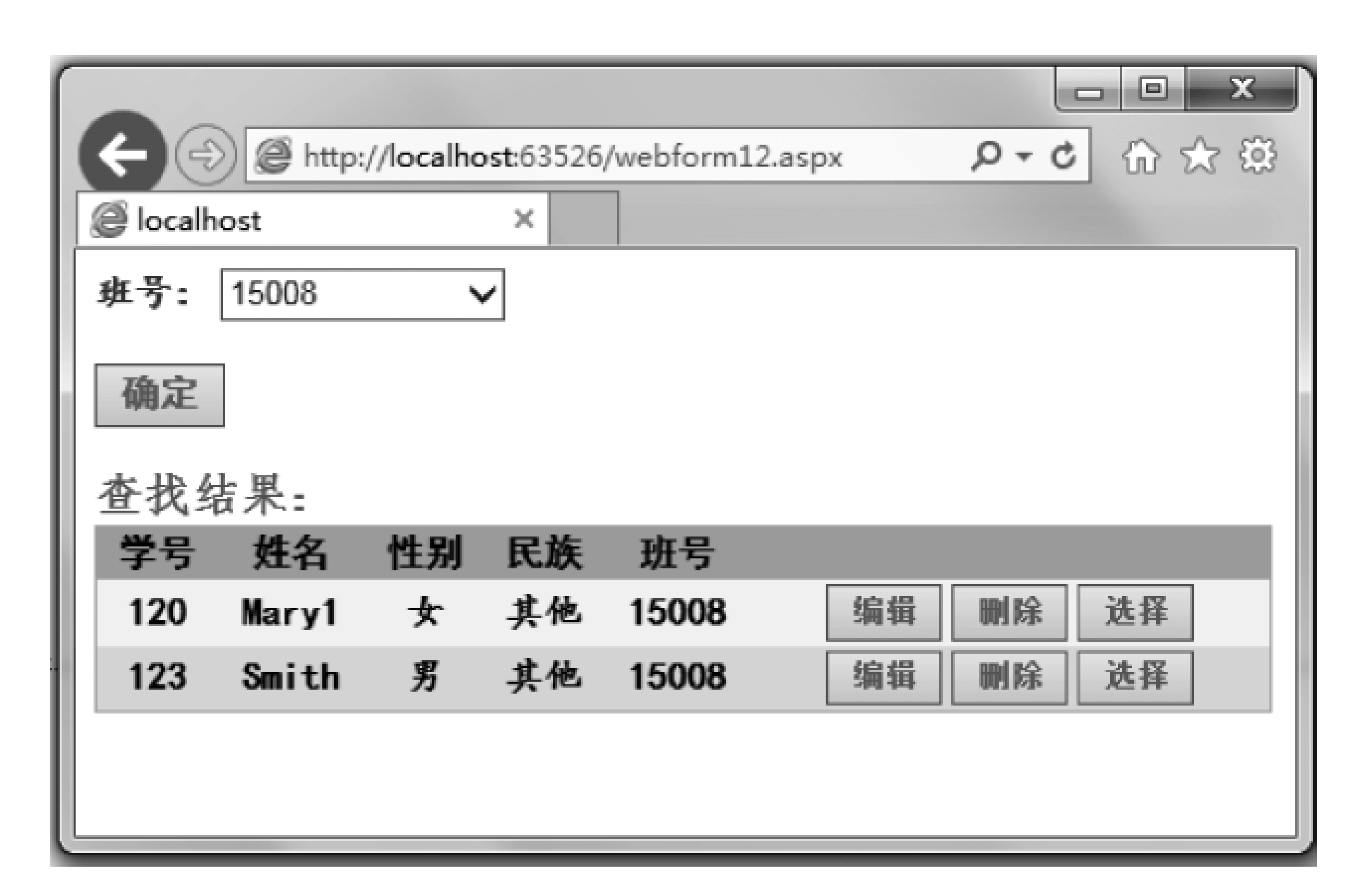


图 11.68　webform12 网页的执行界面五

需要注意的是，在采用自动绑定时(本例采用这种方法)，和 SqlDataSource 控件一样，ObjectDtaSource 控件在更新、新增和删除操作中自动从关联的数据绑定控件(如 GridView 控件)获取参数的集合，这些参数和数据绑定控件的字段具有相同的名称和类型。对于上例，这些参数必须和 StudentDB 类中 SelectData 方法的 SELECT 语句(即“SELECT 学号，姓名，性别，民族，班号 FROM student WHERE 班号＝ @sbh”)的选择字段具有相同的名称和类型，其中学号是 int 类型，其他为 String 类型，否则出现执行错误。

例如，如果将前面 StudentDB 类中 DeleteData 方法改为 DeleteData (int xh)，在执行删除操作时会出现如图 11.69 所示的错误。



图 11.69　网页执行错误一

如果将前面 StudentDB 类中 DeleteData 方法改为 DeleteData (string 学号)，参数赋值改为 mycmd. Parameters. Add("@学号", SqlDbType. VarChar, 5). Value ＝学号。在执行删除操作时会出现如图 11.70 所示的错误。

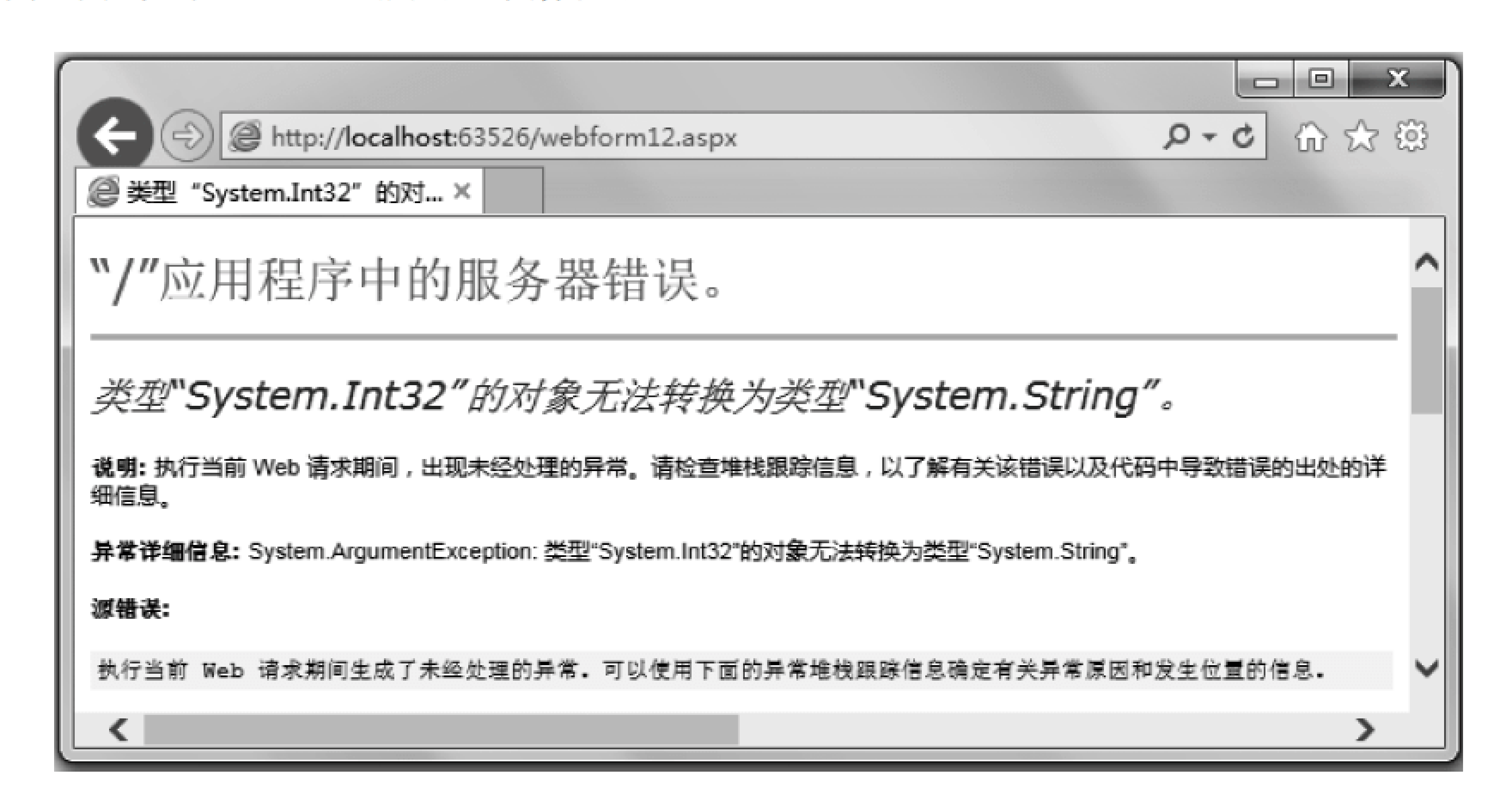


图 11.70　网页执行错误二

归纳起来，SqlDataSource 控件在使用时更容易操作，它直接绑定的是数据库中的表。所以只要绑定好了，对该表的增删改查操作对应的就完成了，可以快速开发网页。其缺点是不可变通，整个程序功能被 SqlDataSource 控件限制了，并且还打乱了 Web 分层体系结构。ObjectDataSource 控件使用起来比较复杂，因为它所对应绑定的是一个类具体方法。例如，查

询对应的是返回值为 DataTable 或 DataSet 的方法，而增删改则对应返回值为 int 型等的方法。这样简化了网页界面上的调用，但是在数据库访问上面还是全部需要用户手动来写的。使用 ObjectDataSource 控件可以完全遵守 Web 分层体系结构，并且方便扩展。

11.7 练习题

1. 单项选择题

（1）以下（　　）不是 ASP .NET 的数据源控件。

A. SqlDataSource　　B. AccessDataSource
C. LinqDataSource　　D. XML

（2）SqlDataSource 可以直接访问（　　）数据库。

A. SQL Server　　B. DB2　　C. Oracle　　D. MySQL

（3）在实现数据绑定时，列表控件主要需要设置 DataSource 和（　　）属性，然后调用 DataBind 方法。

A. DataSourceID　　B. DataTextField
C. Data　　D. DataField

（4）在配置 GridView 控件的 SqlDateSource 数据源控件过程中，单击"高级"按钮的目的是（　　）。

A. 打开其他窗口　　B. 输入新参数
C. 生成 SQL 编辑语句　　D. 优化代码

（5）若要启用 GridView 分页，应将（　　）属性设置 true。

A. AllowSorting　　B. AllowPaging　　C. PageSize　　D. PageIndex

（6）为了在网页中显示多列数据，通常采用（　　）数据绑定控件。

A. DataList　　B. FormView　　C. GridView　　D. ListView

（7）为了在网页中显示单列数据供用户选择，通常采用（　　）数据绑定控件。

A. Table　　B. DropDownList　　C. TexBox　　D. ListBox

（8）GridView 控件不支持的操作是（　　）。

A. 选择　　B. 编辑　　C. 删除　　D. 文件上传

（9）如果希望在 GridView 中显示"上一页"和"下一页"的导航栏，则属性集合 PagerSettings 中的属性 Mode 值应设为（　　）。

A. Numeric　　B. NextPrevious
C. NextPrev　　D. 上一页，下一页

（10）如果对定制后的 GridView 实现排序功能，除设置 GridViewAllowSorting 的属性的值为 True 外，还应该设置（　　）属性。

A. SortExpression　　B. Sort
C. SortField　　D. DataFieldText

（11）DetailsView 控件一次显示（　　）条记录。

A. 1　　B. 2
C. 多　　D. 数据表中全部记录

(12) GridView 控件的属性(　　)用于设置数据源。

A. DataView　B. DataSource　C. DataKey　D. DataValue

(13) ObjectDataSource 控件在进行数据检索时自动调用(　　)属性所设置的方法。

A. SelectMethod　B. InsertMethod

C. UpdateMethod　D. DeleteMethod

(14) ObjectDataSource 控件在进行数据插入时自动调用(　　)属性所设置的方法。

A. SelectMethod　B. InsertMethod

C. UpdateMethod　D. DeleteMethod

(15) ObjectDataSource 控件在进行数据更新时自动调用(　　)属性所设置的方法。

A. SelectMethod　B. InsertMethod

C. UpdateMethod　D. DeleteMethod

(16) ObjectDataSource 控件在进行数据删除时自动调用(　　)属性所设置的方法。

A. SelectMethod　B. InsertMethod

C. UpdateMethod　D. DeleteMethod

(17) 以下是某程序员在一个网页中编写的部分 C# 代码，已知 getdata 是一个返回一个数据集的方法，当某个用户第一次访问该网页时，下列说法正确的是(　　)。

```
private void Page_Load(object sender,System.EventArgs e)
{   if(!Page.IsPostBack)                          //1
    {   DataSet ds = this.getdata ();             //2
        this.GridView1.DataSource = ds.Tables[0]; //3
    }
}
```

A. 用户不能在 GridView 控件中看到数据集中的数据，因为代码行 2、3 不能被执行

B. 用户不能在 GridView 控件中看到数据集中的数据，因为没有设置 GridView 控件的 DataMember 属性

C. 用户不能在 GridView 控件中看到数据集中的数据，因为没有进行数据绑定

D. 用户可以在 GridView 控件中看到数据集中的数据

(18) 对于数据绑定控件，当设置完控件的属性 DataSource 后，需要调用(　　)方法才能显示信息。

A. DataBind()　B. Insert()　C. Update()　D. Select()

2. 问答题

(1) 简述 ASP.NET 数据源控件和数据绑定控件的功能。

(2) 简述 SqlDataSource 控件的功能。

(3) GridView 控件提供了哪些内置功能?

(4) 在 GridView 控件中实现分页需要设置哪些属性?

(5) 简述将 GridView 控件绑定到某个数据源控件，需要设置哪些属性?

(6) 如何在 GridView 控件中添加按钮并实现自己的功能?

(7) 如何在 GridView 控件中启用默认删除功能?

(8) 简述 SqlDataSource 和 ObjectDataSource 控件的不同点。

11.8　上机实验题

在 CH11 网站中添加一个 Exp 网页，其设计界面如图 11.71 所示，用户可以从班号下拉列表 DropDownList1 中选择一个班号，单击“确定”按钮时在 GridView1 控件中显示该班所有学生的平均分，并按平均分递减排序。图 11.72 所示是 15001 班的学生平均分显示结果。要求，DropDownList1 和 GridView1 控件均采用 SqlDataSource 控件作为数据源控件。

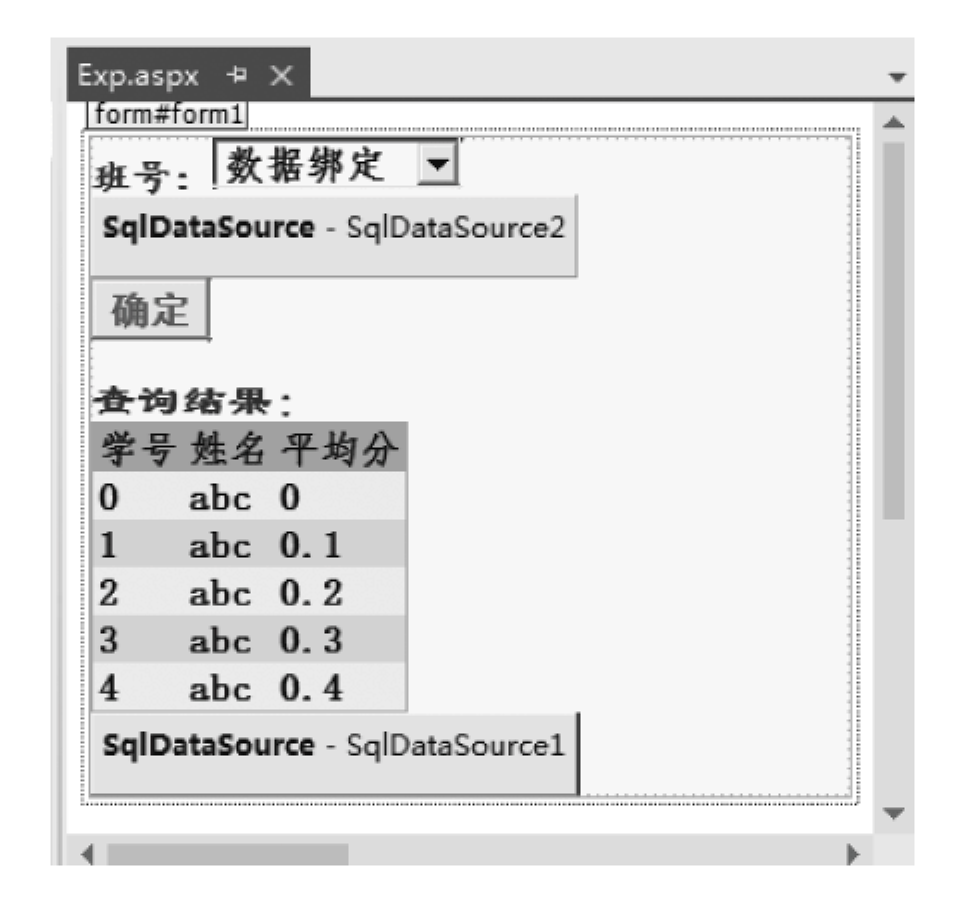


图 11.71　上机实验题网页的设计界面



图 11.72　上机实验题网页的执行界面

第 12 章　电子商务网站开发实例

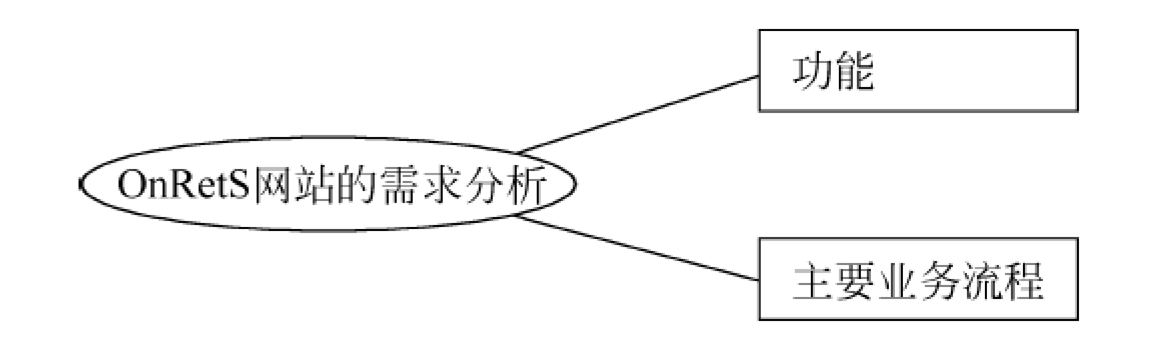


本章指南

- OnRetS网站的需求分析
- OnRetS网站结构设计
- 数据库设计
- 网站公共模块设计
- 主页设计
- 游客功能网页设计
- 顾客功能网页设计
- 管理员功能网页设计
- 操作员功能网页设计

12.1　OnRetS 网站的需求分析

知识梳理



12.1.1　OnRetS 网站的功能

OnRetS 网站是一个简单的在线电子产品销售网站。其分为 4 类用户，如图 12.1 所示。各类用户的主要功能如表 12.1 所示。

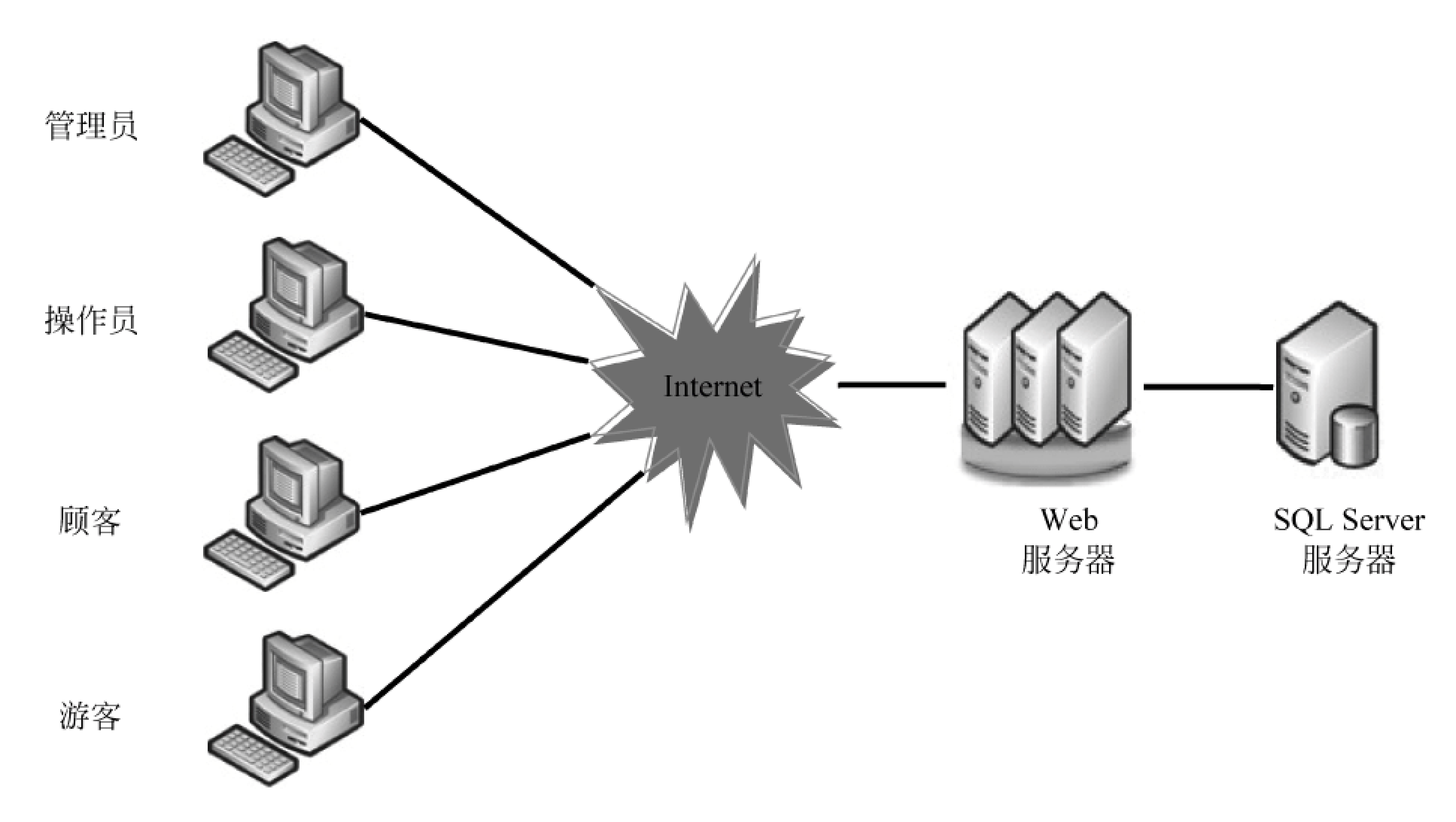


图 12.1　OnretS 网站的 4 类用户

表 12.1　各类用户的功能

用户类型	主功能	子功能
管理员	用户信息管理	添加新用户
		编辑用户信息
	顾客信息管理	查看顾客信息
		临时封杀顾客信息
		查看顾客订单信息
	预警、报警和下架管理	商品库存预警
		商品库存报警
		商品下架
	商品销售统计管理	按商品分类统计
		按商品子类统计
		按商品品牌统计
	设置初始数据	设置顾客学历数据
		设置顾客地区数据
		设置商品类型数据
	我的密码管理	更改我的密码
	系统操作	删除下架的商品信息
		系统初始化
操作员	商品管理	添加新型号商品信息
		更新老商品信息
	我的订单管理	查看新订单
		新订单处理
		新订单结算处理
	我的密码管理	更改密码

续表

用户类型	主　功　能	子　功　能
顾客	购物管理	选购商品放入购物车
		编辑我的购物车
		购物车结算
	订单管理	查看我的订单
		撤销尚未处理的订单
		订单商品评价
	我的信息管理	更改我的信息
		更改我的密码
游客	注册管理	用户注册
	商品管理	查看(浏览)商品

管理员和操作员用户可以维护和管理网站。管理员用户的权限最高，它可以创建其他管理员和操作员。

管理员用户可以设置网站的基础数据，如顾客学历、商品分类、地区层次结构等，可以处理顾客信息、商品预警和报警以及商品销售统计等。

操作员用户维护网站的日常运行，包括商品的上架、顾客订单处理和结算。

游客用户只能浏览网页的所有有效商品信息，可以通过注册变为顾客用户。

顾客用户可以购物和进行订单管理，包括购物车结算、订单撤销和商品评价等。

本网站不支持订单在线支付，采用货到付款的方式。顾客下订单后，操作员处理完订单，产生订单表，交由快递员发货，顾客收到商品并支付购物款后，快递员告诉操作员，操作员进行该订单的结算处理，一次购物过程结束。

12.1.2　OnRetS 网站的主要业务流程

网站的实体有各类用户、购物车、商品库、订单库、销售库和顾客库等。本节给出主要业务流程。

(1) 游客注册流程

游客注册流程如图 12.2 所示，仅仅涉及顾客库，即把个人信息存入顾客库中。

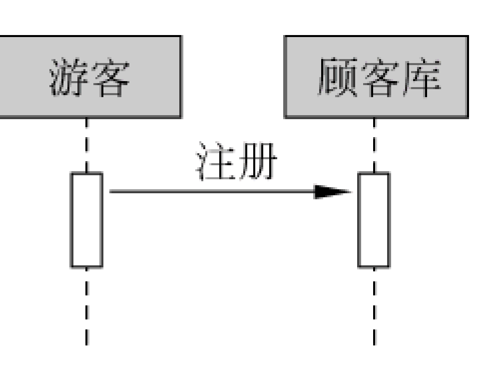


图 12.2　游客注册流程

(2) 顾客购物流程

顾客购物流程如图 12.3 所示。其中购物结算流程如图 12.4 所示。

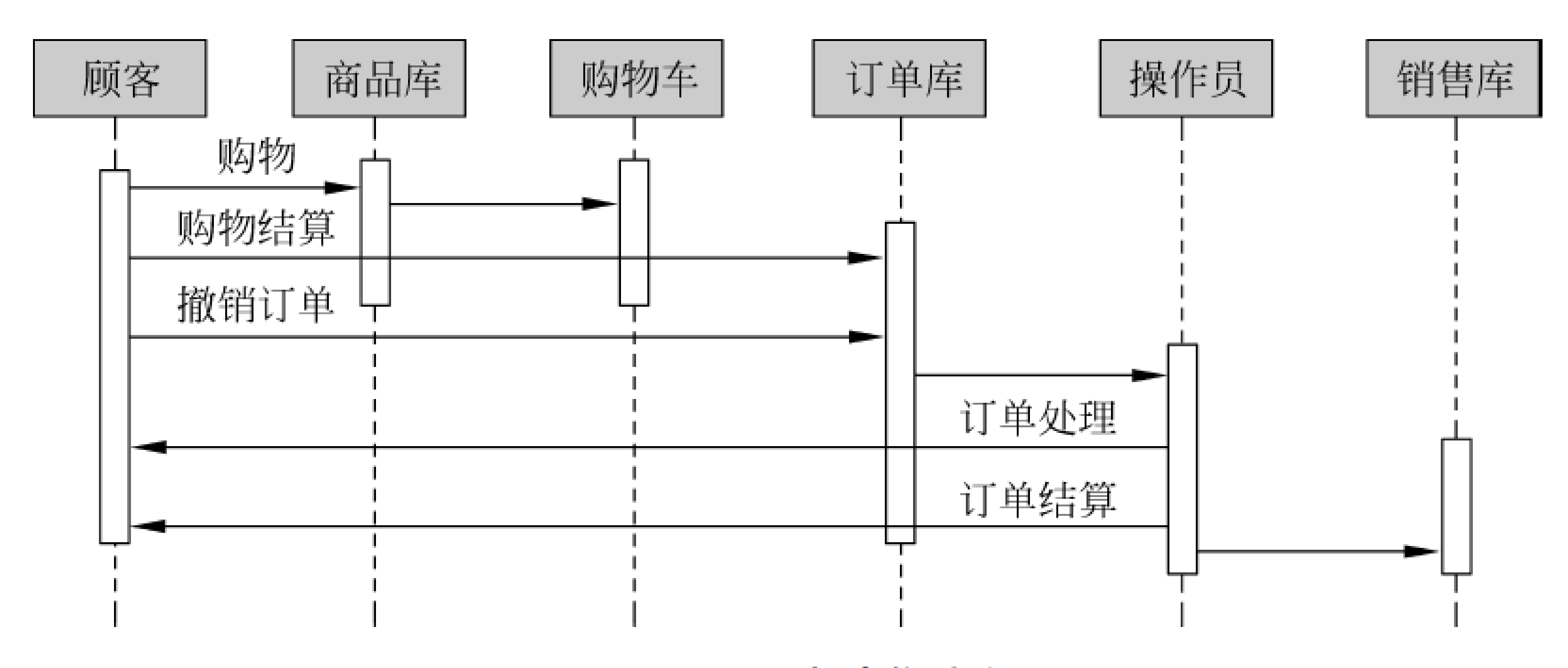


图 12.3　顾客购物流程

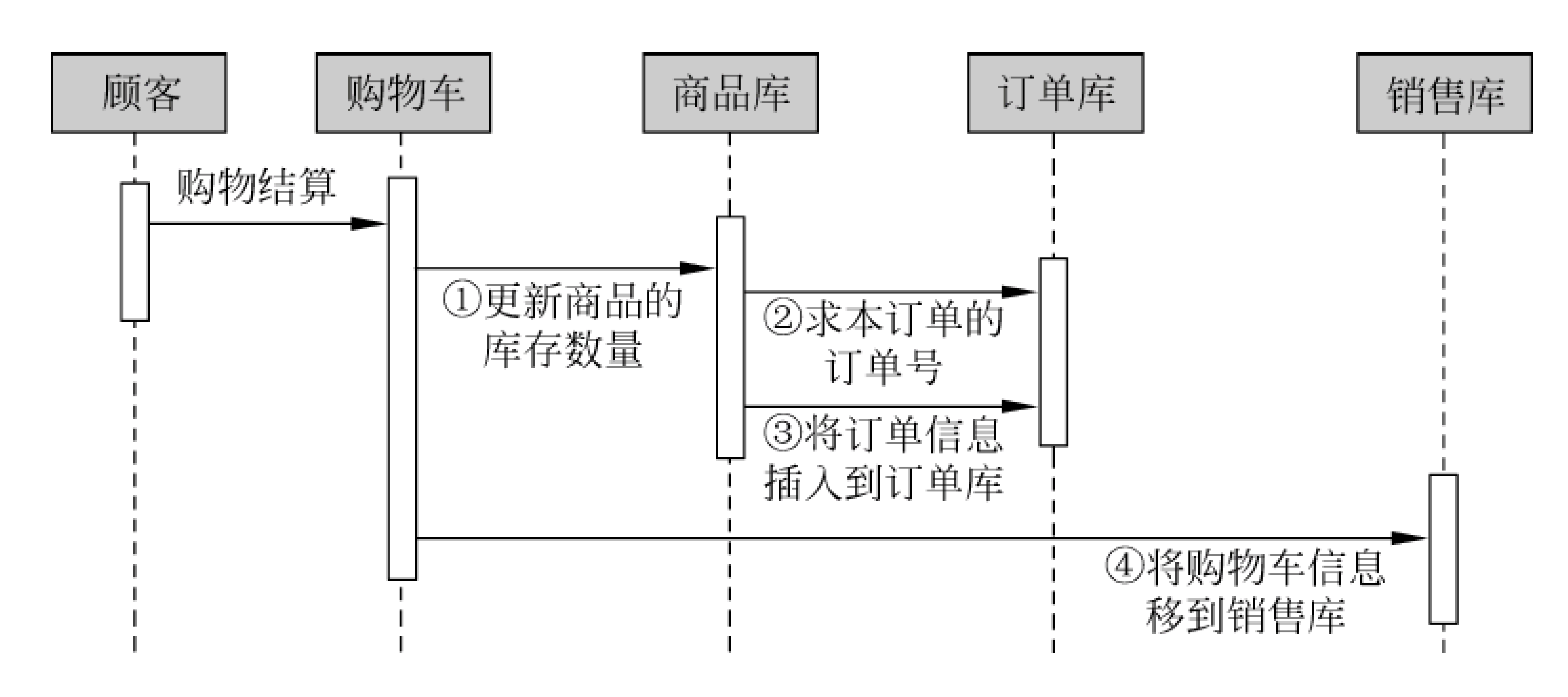


图 12.4　顾客结算商品的流程

(3) 管理员设置基础数据流程

管理员设置基础数据流程如图 12.5 所示，基础信息库包括顾客学历、商品分类和地区层次结构等。

(4) 操作员商品入库操作流程

操作员商品入库流程如图 12.6 所示，其中涉及基础信息库的商品分类信息。

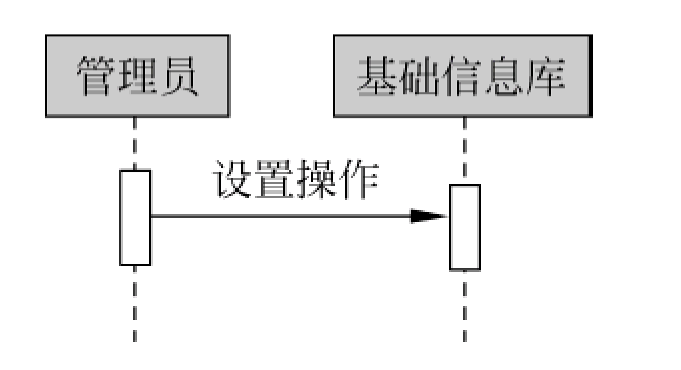


图 12.5　管理员设置基础数据流程

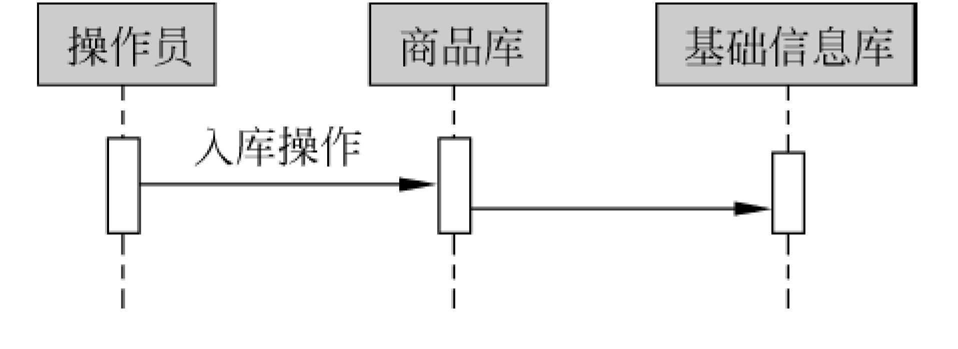
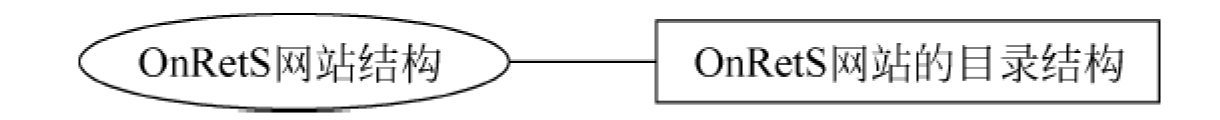


图 12.6　操作员商品入库流程

其他流程详见后面各功能的网页设计。

12.2　OnRetS 网站结构设计

知识梳理



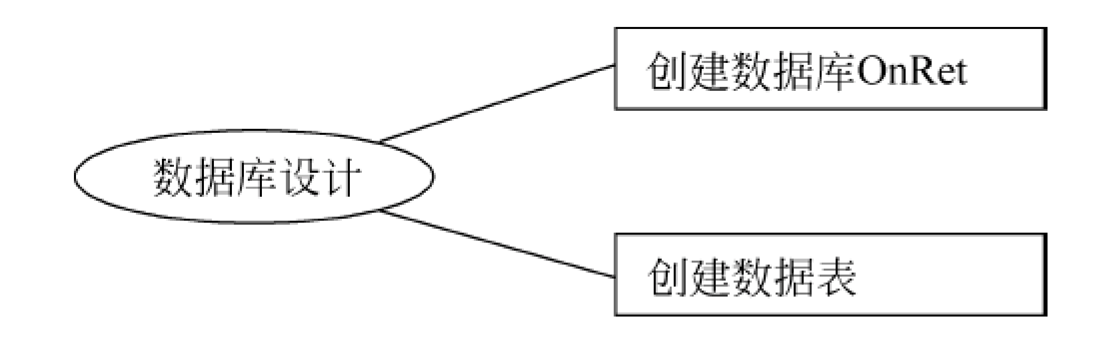
OnRetS 网站的结构见图 3.1，对应的物理路径为 D:\OnRetS，其中包括所有数据库、图像和网页文件类存放。各子目录的说明如下：

- App_Data 目录：用于存放 OnRet 数据库文件和日志文件。
- App_Code 目录：存放数据库访问类文件 CommDB.cs。
- images 目录：存放网页设计中使用的一些图像文件(top.jpg、bottom.jpg 等)。
- App_Themes 目录：存放主题 Blue 和样式文件 StyleSheet.css。
- Picture 目录：存放商品的图像文件(通常文件名与商品编号一致，如编号为 1111 的商品对应的图像文件为 1111.jpg)
- Manager 目录：存放管理员的主要网页文件。
- Operator 目录：存放操作员的主要网页文件。
- Customer 目录：存放游客的主要网页文件。

- Tourist 目录：存放顾客的主要网页文件。
- 根目录：存放 web. config(配置文件)、Default. aspx(主页文件)、MasterPage. master(母版页文件)、dispinfo. aspx(公共信息显示网页文件)、managermenu. aspx(管理员主页文件)、operatormenu. aspx(操作员主页文件)、customermenu. aspx(顾客主页文件)和 touristmenu. aspx(游客主页文件)。

12.3　数据库设计

知识梳理



12.3.1　创建数据库 OnRet

首先，使用 Visual Studio 2012 创建一个文件系统的空网站 OnRetS，其位置为 D:\OnRetS 目录，建立一个名称为 App_Data 的子目录。

采用“Window 身份验证”模式启动 SQL Server 2012 数据库引擎，选择“数据库|新建数据库”目录，创建 OnRet 数据库的对话框如图 12.7 所示，数据文件存放在 D:\OnRetS\App_Data 目录中。单击“确定”按钮便创建了 OnRet 数据库。



图 12.7　创建 OnRet 数据库

12.3.2　创建数据表

OnRet 数据库中共创建如下 10 个数据表。

(1) User 为用户信息表,用于存放管理员和操作员的信息。其表结构如图 12.8 所示。根管理员为 system/manager,管理员可以让其他操作员变为无效的。无效操作员不能登录本网站。

LCB-PC.OnRet - dbo.Users

列名	数据类型	允许 Null 值
用户名	char(20)	☐
密码	char(10)	☑
类型	char(10)	☑
有效否	bit	☑
		☐

图 12.8　User 表结构

(2) Customer 为顾客信息表,用于存放所有顾客的信息。其表结构如图 12.9 所示。管理员可以让每个顾客变为无效的,无效顾客不能登录本网站。

(3) Products 为商品信息表,用于存放所有商品的信息。其表结构如图 12.10 所示。管理员可以下架某个商品,这样对应的商品信息变为无效,无效商品不能被顾客购买。该表中"图片"列是字符串类型的,存放商品对应的图像文件名,而不是实际图像文件。例如,Products 表中部分商品信息如图 12.11 所示,从中看到,编号为 1111 的商品对应的图像文件为"~//Picture//1111.jpg",即网站子目录 Picture 中的 1111.jpg 文件。

LCB-PC.OnRet - dbo.Customers

列名	数据类型	允许 Null 值
用户名	char(20)	☐
密码	char(10)	☑
姓名	char(20)	☑
学历	char(10)	☑
年龄	int	☑
地区	char(10)	☑
省份	char(10)	☑
市	char(10)	☑
县	char(10)	☑
住址	char(40)	☑
邮箱	char(40)	☑
电话	char(20)	☑
有效否	bit	☑
		☐

图 12.9　Customer 表结构



LCB-PC.OnRet - dbo.Products

列名	数据类型	允许 Null 值
商品编号	char(20)	☐
分类	char(20)	☑
子类	char(20)	☑
品牌	char(20)	☑
型号	char(20)	☑
单价	float	☑
库存数量	int	☑
图片	char(50)	☑
有效否	bit	☑
星数	float	☑
评论数	int	☑
		☐

图 12.10　Products 表结构

	商品编号	分类	子类	品牌	型号	单价	库存数量	图片	有效否	星数	评论数
1	1111	手机/数码	手机	小米	红米手机2	749	194	~//Picture//1111.jpg	1	0	0
2	1112	手机/数码	手机	小米	红米note	1067	95	~//Picture//1112.jpg	1	0	0
3	1121	手机/数码	手机	华为	华为P8	2888	82	~//Picture//1121.jpg	1	0	0
4	1122	手机/数码	手机	华为	荣耀6Plus	1999	78	~//Picture//1122.jpg	1	0	0

图 12.11　Products 表中部分记录

(4) Comment 为商品信息表,用于存放所有顾客对商品的评价信息。其表结构如图 12.12 所示。

(5) Area 为地区层次结构表,用于商品销售分析以及顾客地区结构分析。其表结构如图 12.13 所示。其中,"编号"列属性如图 12.14 所示,它是一个标识规范列,其值是自动增长的。

LCB-PC.OnRet - dbo.Comment

列名	数据类型	允许 Null 值
商品编号	char(20)	☑
用户名	char(20)	☑
评语	char(200)	☑
分数	int	☑
		☐

图 12.12 Comment 表结构



LCB-PC.OnRet - dbo.Area

列名	数据类型	允许 Null 值
编号	int	☐
地区	char(10)	☑
省份	char(10)	☑
市	char(10)	☑
县	char(10)	☑
		☐

图 12.13 Area 表结构

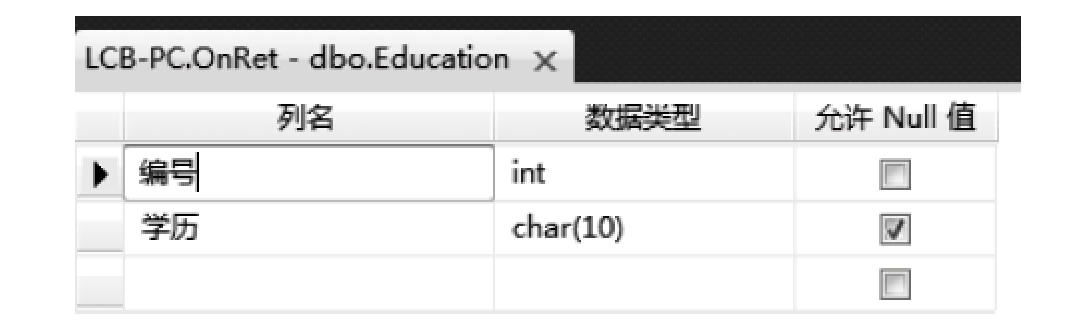
(6) Education 为顾客学历表，用于顾客学历结构分析。其表结构如图 12.15 所示。其中“编号”列也是标识规范列，其值是自动增长的。



列属性

(常规)	
(名称)	编号
默认值或绑定	
数据类型	int
允许 Null 值	否
表设计器	
RowGuid	否
标识规范	是
(是标识)	是
标识增量	1
标识种子	1
不用于复制	否
大小	4
计算列规范	
简洁数据类型	int

图 12.14 Area 表列“编号”的属性



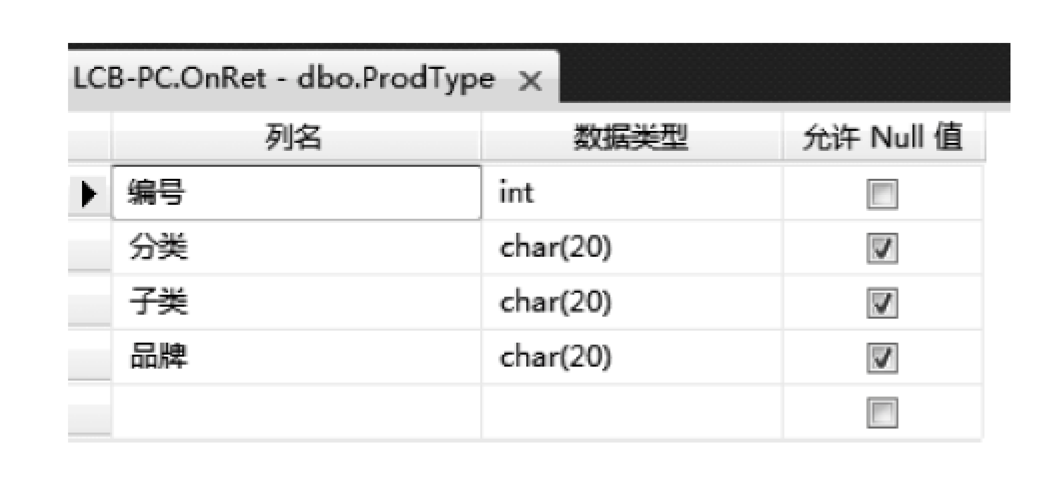
LCB-PC.OnRet - dbo.Education

列名	数据类型	允许 Null 值
编号	int	☐
学历	char(10)	☑
		☐

图 12.15 Education 表结构

(7) ProdType 为商品分类表，用于商品分类结构分析。其表结构如图 12.16 所示。其中“编号”列也是标识规范列，其值是自动增长的。

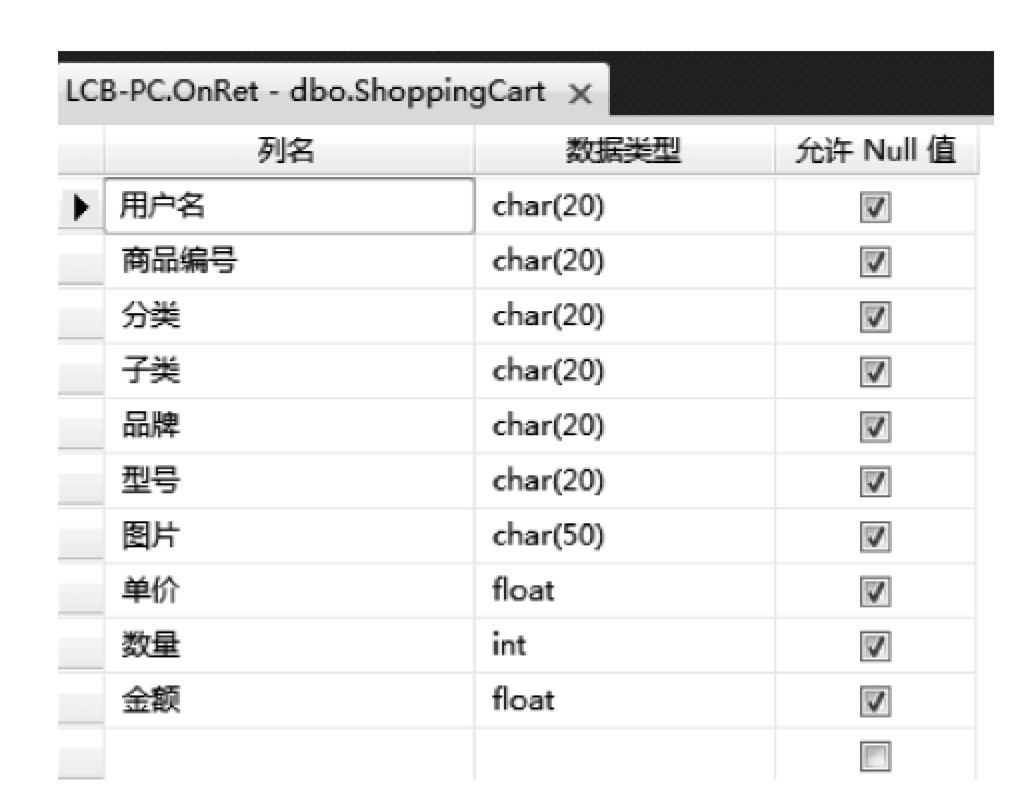
(8) ShoppingCart 为商品购物车表，用于存放顾客放入购物车的商品信息。其表结构如图 12.17 所示。



LCB-PC.OnRet - dbo.ProdType

列名	数据类型	允许 Null 值
编号	int	☐
分类	char(20)	☑
子类	char(20)	☑
品牌	char(20)	☑
		☐

图 12.16 ProdType 表结构



LCB-PC.OnRet - dbo.ShoppingCart

列名	数据类型	允许 Null 值
用户名	char(20)	☑
商品编号	char(20)	☑
分类	char(20)	☑
子类	char(20)	☑
品牌	char(20)	☑
型号	char(20)	☑
图片	char(50)	☑
单价	float	☑
数量	int	☑
金额	float	☑
		☐

图 12.17 ShoppingCart 表结构

(9) OrderForm 为商品订单表，用于存放所有顾客的订单信息。其表结构如图 12.18 所示。

（10）Sales 为商品销售表，用于存放所有商品销售信息。其表结构如图 12.19 所示。

LCB-PC.OnRet - dbo.OrderForm

列名	数据类型	允许 Null 值
订单号	int	☑
日期	date	☑
用户名	char(20)	☑
姓名	char(20)	☑
地区	char(10)	☑
省份	char(10)	☑
市	char(10)	☑
县	char(10)	☑
住址	char(40)	☑
邮箱	char(40)	☑
电话	char(20)	☑
总数量	int	☑
总金额	float	☑
处理否	bit	☑
结算否	bit	☑
		☐

图 12.18　OrderForm 表结构

LCB-PC.OnRet - dbo.Sales

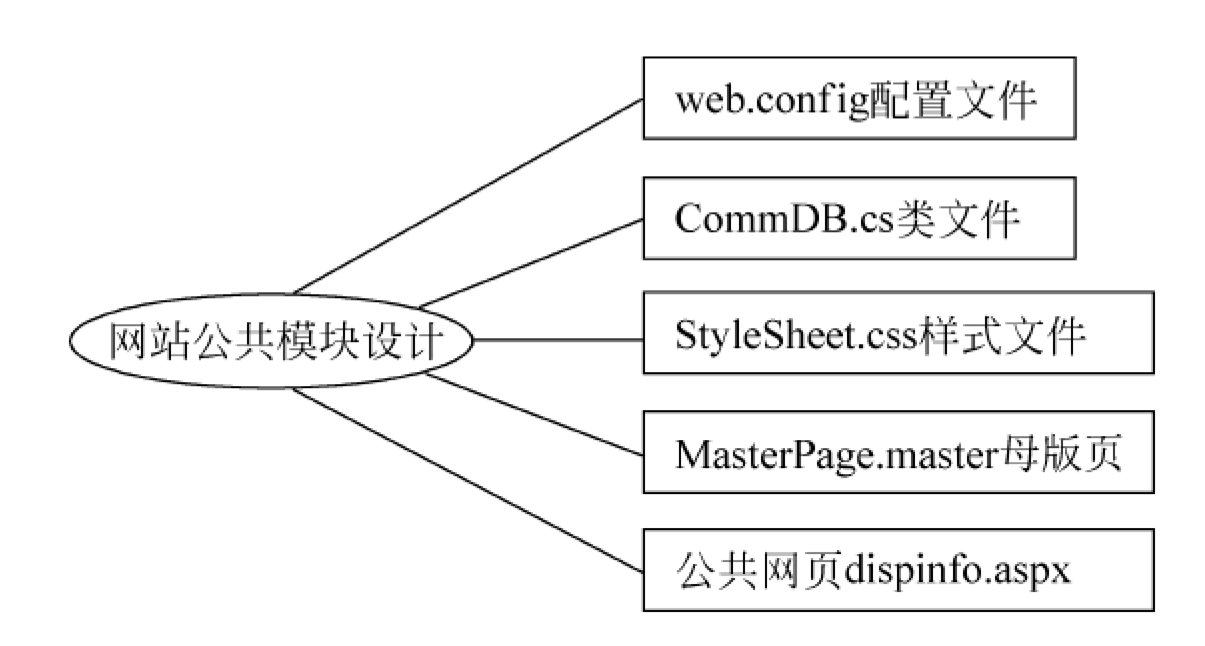
列名	数据类型	允许 Null 值
订单号	int	☑
日期	date	☑
用户名	char(20)	☑
商品编号	char(20)	☑
分类	char(20)	☑
子类	char(20)	☑
品牌	char(20)	☑
型号	char(20)	☑
单价	float	☑
数量	int	☑
金额	float	☑
		☐

图 12.19　Sales 表结构

上述表中，包含了一些冗余字段，所谓冗余字段是指这些信息可以从其他表通过表连接得到。为了减轻服务器的负担，这样的设计是合适的。

12.4　网站公共模块设计

知识梳理



公共功能模块由 web.config、CommDB.cs、StyleSheet.css、MasterPage.master（母版页文件）和 dispinfo.aspx 组成。

12.4.1　web.config 配置文件

本网站的 web.config 配置文件十分简单，由于没有采用 ASP .NET 的登录功能，因此不需配置提供程序，只增加连接字符串节和设置 Default.aspx 为主页，存放在网站的根目录下。该配置文件的代码如下：

```
<?xml version = "1.0"?>
<configuration>
  <appSettings>
```

```
      <add key = "ValidationSettings:UnobtrusiveValidationMode" value = "None" />
    </appSettings>
    <connectionStrings>
      <add name = "myconnstring"
          connectionString = "Data Source = LCB - PC;Initial Catalog = OnRet;
             Integrated Security = True"
          providerName = "System.Data.SqlClient" />
    </connectionStrings>
    <system.web>
      <authentication mode = "Forms">
        <forms loginUrl = "Default.aspx"/>
      </authentication>
      <compilation debug = "true" targetFramework = "4.5">
      </compilation>
      <httpRuntime targetFramework = "4.5"/>
    </system.web>
</configuration>
```

12.4.2　CommDB.cs 类文件

该类文件包括通用数据库访问方法和随机产生验证码方法等，被其他网页引用，存放在网站的 App_Code 目录中。文件代码如下：

```
using System;
using System.Data;
using System.Data.SqlClient;
public class CommDB
{   public CommDB()                              //默认构造函数
    {}
    //**********************************************************************
    //返回 SELECT 语句执行后记录集中的行数
    //**********************************************************************
    public int Rownum(string sql)
    {                                            //sql 参数指出 SQL 语句
        int i = 0;
        string mystr  =  System.Configuration.ConfigurationManager.
            ConnectionStrings["myconnstring"].ToString();
        //从 web.config 文件获取连接字符串
        SqlConnection myconn  =  new SqlConnection();
        myconn.ConnectionString  =  mystr;
        myconn.Open();
        SqlCommand mycmd  =  new SqlCommand(sql, myconn);
        SqlDataReader myreader  =  mycmd.ExecuteReader();
        while (myreader.Read())                  //循环读取信息
        {   i++; }
        myconn.Close();
        return i;                                //返回读取的行数
    }
    //**********************************************************************
    //执行 SQL 语句,返回是否成功执行.SQL 语句最好是如下:
    //UPDATE 表名 SET 字段名 = value,字段名 = value WHERE 字段名 = value
    //DELETE FROM 表名 WHERE 字段名 = value
    //INSERT INTO 表名 (字段名,字段名) values (value,value)
    //**********************************************************************
    public Boolean ExecuteNonQuery(string sql)
```

```
{    string mystr = System.Configuration.ConfigurationManager.
     ConnectionStrings["myconnstring"].ToString();
     SqlConnection myconn = new SqlConnection();
     myconn.ConnectionString = mystr;
     myconn.Open();
     SqlCommand mycmd = new SqlCommand(sql,myconn);
     try
     {    mycmd.ExecuteNonQuery();
          myconn.Close();
     }
     catch
     {    myconn.Close();
          return false;
     }
     return true;
}
//*********************************************************************
//执行 SELECT 语句,返回 DataSet 对象
//*********************************************************************
public DataSet ExecuteQuery(string sql,string tname)
{    string mystr = System.Configuration.ConfigurationManager.
          ConnectionStrings["myconnstring"].ToString();
     SqlConnection myconn = new SqlConnection();
     myconn.ConnectionString = mystr;
     myconn.Open();
     SqlDataAdapter myda = new SqlDataAdapter(sql,myconn);
     DataSet myds = new DataSet();
     myda.Fill(myds,tname);
     myconn.Close();
     return myds;
}
//*********************************************************************
//执行 SELECT 语句,返回聚合函数结果
//*********************************************************************
public string ExecuteAggregateQuery(string sql)
{    string jg;
     string mystr = System.Configuration.ConfigurationManager.
          ConnectionStrings["myconnstring"].ToString();
     SqlConnection myconn = new SqlConnection();
     myconn.ConnectionString = mystr;
     myconn.Open();
     SqlCommand mycmd = new SqlCommand();
     mycmd.CommandText = sql;
     mycmd.Connection = myconn;
     jg = mycmd.ExecuteScalar().ToString();
     myconn.Close();
     return jg;
}
//*********************************************************************
/// 实现随机验证码:返回生成的随机数
//*********************************************************************
public string RandomNum(int n)                    //n 为验证码的位数
{    //定义一个包括数字、大写英文字母和小写英文字母的字符串
     string strchar = "0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,G,H," +
          "I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,";
```

```
        //将 strchar 字符串转化为数组
        //String.Split 方法返回包含此实例中的子字符串的 String 数组.
        string[] arry = strchar.Split(',');
        string num = "";
        //记录上次随机数值,尽量避免产生几个一样的随机数
        int temp = -1;
        //采用一个简单的算法以保证生成随机数的不同
        Random rand = new Random();
        for (int i = 1; i < n + 1; i++)
        {   if (temp != -1)
            {   //unchecked 关键字用于取消整型算术运算和转换的溢出检查
                //DateTime.Ticks 属性获取表示此实例的日期和时间的刻度数
                rand = new Random(i * temp * unchecked((int)DateTime.Now.Ticks));
            }
            //Random.Next 方法返回一个小于所指定最大值的非负随机数
            int t = rand.Next(35);
            if (temp != -1 && temp == t)
                return RandomNum(n);
            temp = t;
            num += arry[t];
        }
        return num;                          //返回生成的随机数
    }
}
```

12.4.3　StyleSheet.css 样式文件

该文件包含一些样式定义，被其他网页引用，存放在网站的 App_Themes 目录中。文件代码如下：

```
.auto-stringstyle                         /*输入文本框提示文字样式*/
{   font-family: 楷体; font-size: medium;
    color: #0000FF; font-weight: bold;
    text-align:right; height:22px;
}
.auto-captionstyle                        /*标题样式*/
{   font-size: 16pt; color: #ff0099;
    font-family: 幼圆; font-weight: bold;
    text-align:center; height:40px;
}
.auto-resettyle                           /*重置按钮样式*/
{   font-weight: bold; color: red;
    font-family: 黑体; font-size: medium;
}
a:visited                                 /*定义超链接被访问过后的显示颜色*/
{   text-decoration:none;
    color:#0000FF;font-weight:bold;
}
a:link                                    /*定义正常显示的超链接颜色*/
{   text-decoration:none;
    color:#FF6A00;font-weight:bold;
}
#tablecenter                              /*表格居中样式*/
{   margin-left: auto; margin-right: auto;
    vertical-align: middle; background-color: #99ccff;
    width: 519px;
}
```

12.4.4　MasterPage.master 母版页

母版页 MasterPage.master 存放在网站根目录中。该母版页中包含一个 3×3 的表格，第 1 行放置 images/top.jpg 图形文件，第 3 行放置 images/bottom.jpg 图形文件，第 2 行第 1 列和第 3 列各放置一个 images/edges.jpg 图形文件，第 2 行第 2 列放置一个 ContentPlaceHolder 控件 ContentPlaceHolder1。其设计界面如图 12.20 所示。



图 12.20　MasterPage.master 设计界面

在本网站中，MasterPage.master 作为 Default.aspx 和各种用户主页的母版页，这样达到统一网页设计界面的目的。

12.4.5　公共网页 dispinfo.aspx

该网页用于在各种用户主页中显示用户操作的提示信息，它存放在网站的根目录下。

dispinfo.aspx 网页的设计界面如图 12.21 所示，其中只有一个标签 Label1，包含的 C＃代码如下：

```
using System;
public partial class Welcome : System.Web.UI.Page
{    protected void Page_Load(object sender, EventArgs e)
     {
          Label1.Text = Request.QueryString["info"];
     }
}
```

其他调用它的网页向它传递 info 字符串，然后在 Label1 中显示。例如，以下调用语句显示的结果如图 12.22 所示。

```
dispinfo.aspx?info = 欢迎使用本系统"
```

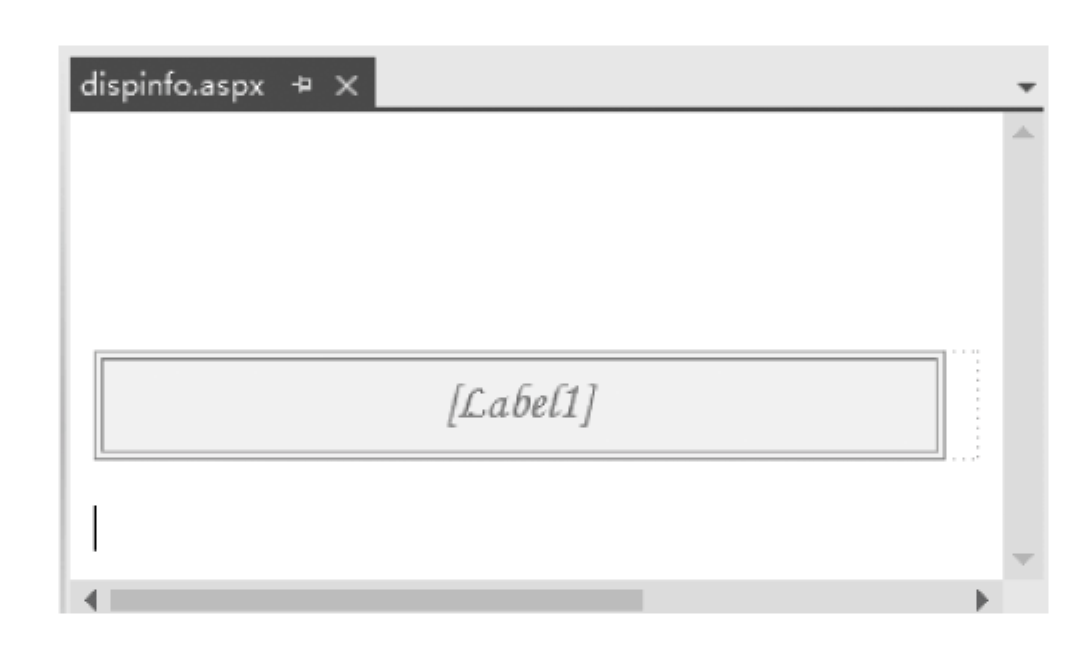


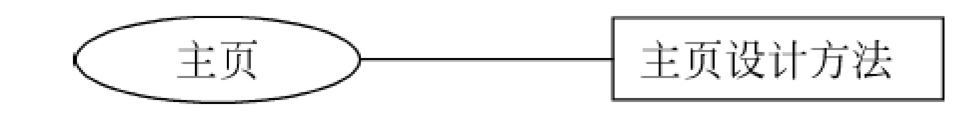
图 12.21　dispinfo 网页的设计界面

欢迎使用本系统

图 12.22　dispinfo 网页的执行界面

12.5　主页设计

知识梳理



本网站的主页是 Default.aspx。它提供用户登录功能。

Default 网页的设计界面如图 12.23 所示，其母版页为 MasterPage.master，在 Contenl 中包含一个 6×2 的表格，表格中主要有用户编号文本框 TextBox1，密码文本框 TextBox2，用户类型单选按钮(RadioButton1、RadioButton2 和 RadioButton3)，输入验证码文本框 TextBox3，显示验证码标签 Label1、"登录"按钮 Button1、"重置"按钮 Button2、"看不清"按钮 Button3 和"游客入口"按钮 Button4。



图 12.23　Default 主页的设计界面

主页对应的 C# 代码如下：

```
using System;
using System.Web.UI;
public partial class _Default : System.Web.UI.Page
{   CommDB mydb = new CommDB();                        //创建 CommDB 类对象
    protected void Page_Load(object sender, EventArgs e)
    {   if (!Page.IsPostBack)                          //在网页首发时执行
            Label1.Text = mydb.RandomNum(4);
        else                                           //在回传时保持密码
            TextBox2.Attributes.Add("value", TextBox2.Text);
    }
    protected void Button1_Click(object sender, EventArgs e)
    {   string mysql;                                  //SQL 语句
        int i;
        if (TextBox1.Text.Trim() == "" || TextBox2.Text.Trim() == "")
```

```
        {   Response.Write("<script>alert('用户名和密码不能为空!')</script>");
            return;
        }
        if (TextBox3.Text.ToUpper().Trim()!= Label1.Text.Trim())          //若验证码输入错误
            Response.Write("<script>alert('你的验证码输入错误,"
                + "请重输入!')</script>");
        else                                            //若验证码输入正确
        {   if (RadioButton1.Checked)                   //顾客登录
            {   mysql = "SELECT 用户名 FROM Customers WHERE 用户名 = '"
                    + TextBox1.Text + "' AND 密码 = '" + TextBox2.Text
                    + "' AND 有效否 = '1'";
                i = mydb.Rownum(mysql);                 //执行 SQL 语句并返回行数 i
                if (i > 0)                              //合法顾客
                {   Session["uname"] = TextBox1.Text.Trim();             //保存顾客用户名
                    Server.Transfer("~/customermenu.aspx");              //转向顾客主页
                }
                else                                    //非法用户
                    Response.Write("<script>alert('对不起,你输入的"
                        + "用户名/密码错误或者已无效,请查实!')</script>");
            }
            else if (RadioButton2.Checked)              //操作员登录
            {   mysql = "SELECT 用户名 FROM Users WHERE 用户名 = '" + TextBox1.Text
                  + "' AND 密码 = '" + TextBox2.Text + "'"
                  + " AND 类型 = '操作员' AND 有效否 = '1'";
                i = mydb.Rownum(mysql);                 //执行 SQL 语句并返回行数 i
                if (i > 0)                              //合法操作员用户
                {   Session["uname"] = TextBox1.Text.Trim();             //保存操作员用户名
                    Server.Transfer("~/operatormenu.aspx");              //转向操作员主页
                }
                else                                    //非法用户
                    Response.Write("<script>alert('对不起,你输入的"
                      + "用户名/密码错误或者已无效,请查实!')</script>");
            }
            else if (RadioButton3.Checked)              //管理员登录
            {   mysql = "SELECT 用户名 FROM Users WHERE 用户名 = '" + TextBox1.Text
                  + "' AND 密码 = '" + TextBox2.Text + "' AND 类型 = '管理员'";
                i = mydb.Rownum(mysql);                 //执行 SQL 语句并返回行数 i
                if (i > 0)                              //合法管理员用户
                {   Session["uname"] = TextBox1.Text.Trim();             //保存管理员用户名
                    Server.Transfer("~/managermenu.aspx");               //转向管理员主页
                }
                else                                    //非法用户
                    Response.Write("<script>alert('对不起,你输入的"
                      + "用户名或者密码错误,请查实!')</script>");
            }
        }
    }
    protected void Button3_Click(object sender, EventArgs e)
    {
        Label1.Text = mydb.RandomNum(4);                //获取验证码并显示在 Label1 控件中
    }
    protected void Button4_Click(object sender, EventArgs e)
    {   Session["uname"] = "游客";                      //保存"游客"用户名
        Server.Transfer("~/touristmenu.aspx"); //转向游客主页
    }
}
```

本网页用 Session("uname")会话保存登录用户的用户名，在后面调用的网页中多次使用。

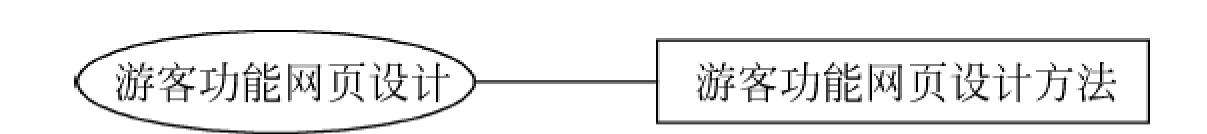
本主页在 IE 浏览器中的执行界面如图 12.24 所示，这里是管理员登录，输入用户名和密码，选择“管理员”类型，输入正确的验证码，单击“登录”按钮即可进入管理员主页(管理员菜单网页)。如果是游客，只需单击“游客入口”按钮即可进入游客主页。



图 12.24　Default 主页的执行界面

12.6　游客功能网页设计

知识梳理



12.6.1　游客功能主页设计

当游客进入网站后，首先显示游客功能主页 touristmenu，其设计界面如图 12.25 所示。它是基于母版页 MasterPage.master 的。该网页位于网站根目录下，它所调用的所有网页都放置在子 Tourist 目录中。

在 Content1 中包含一个 2×2 的表格，表格的第 1 行两列合并，放置一个提示标签 Label1。

表格的第 2 行第 1 列放置一个 TreeView1 控件和一个超链接 HyperLink1。TreeView1 控件的源视图代码如下：

```
<asp:TreeView ID = "TreeView1" runat = "server" style = "text - align:center;
    font - family:仿宋;font - weight:bold;font - size: 16px">
    <Nodes>
      <asp:TreeNode Text = "注册管理" Value = "注册管理"
          NavigateUrl = "dispinfo.aspx?info = 欢迎使用本系统" Target = "Iframe1">
          <asp:TreeNode Text = "用户注册" Value = "用户注册"
            NavigateUrl = "~/Tourist/Registered.aspx" Target = "Iframe1">
          </asp:TreeNode>
      </asp:TreeNode>
```

```
        <asp:TreeNode Text = "商品管理" Value = "购物管理"
              NavigateUrl = "dispinfo.aspx?info = 欢迎使用本系统" Target = "Iframe1">
           <asp:TreeNode Text = "查看(浏览)商品" Value = "查看(浏览)商品"
              NavigateUrl = "～/Tourist/QueryProduct.aspx" Target = "Iframe1">
           </asp:TreeNode>
        </asp:TreeNode>
      </Nodes>
</asp:TreeView>
```

TreeView1 控件起到菜单的作用，当用户单击其中叶子节点项时，调用 NavigateUrl 属性所指定的网页，被调用的网页均在 Iframe1 框架中显示。当用户单击非叶子节点项时，统一调用 dispinfo 网页显示提示信息。



图 12.25　游客主页 touristmenu 的设计界面

超链接 HyperLink1 用于返回到网站主页，对应的源视图代码如下：

```
<asp:HyperLink ID = "HyperLink1" runat = "server"
   style = "font - family:黑体; font - weight:bold;font - size:16px;color:#009900"
   NavigateUrl = "～/Default.aspx" Target = "_self">退出本系统
</asp:HyperLink>
```

表格的第 2 行第 2 列放置一个 IFrame 框架，其源视图代码如下：

```
<iframe name = "Iframe1" id = "Iframe1"
    style = " height:480px; width:99 % ;text - align:center"
    src = "dispinfo.aspx?info = 欢迎使用本系统">
</iframe>
```

Iframe1 框架中初始显示 dispinfo 网页，以后显示游客操作所调用的网页。由于 Visual Studio 中没有 IFrame 控件，这里采用直接输入源视图代码的方式添加 Iframe1 框架。

本网页对应的 C# 代码如下：

```
using System;
public partial class touristmenu : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Text = "游客端→欢迎你:" + Session["uname"];
    }
}
```

游客进入本网站后的执行界面如图 12.26 所示。用户可以单击 TreeView1 控件的节点执行相应的功能。



图 12.26　游客主页 touristmenu 的执行界面

12.6.2　“用户注册”功能网页设计

用户注册网页 Registered 实现游客的注册操作，其设计界面如图 12.27 所示，其中包含一个 15 行 4 列的表格。第 4 列主要是验证控件。为了用户操作方便，如学历、地区、省份、市和县这类基础数据，用户可以从对应的下拉列表中选择，而且必须选择一个非空项。

学历下拉列表为 DropDownList1，它与 Education 表绑定。在 Page_Load 事件处理过程中就进行绑定，在这个网页执行中绑定的值不发生改变。

地区下拉列表为 DropDownList2，它与 Area 表绑定。当用户选择一个地区后，省份下拉列表 DropDownList3 的数据应随之发生改变，如当用户选择“华东”地区时，DropDownList3 下拉列表中只列出华东地区的省份。为此，在 DropDownList2 下拉列表上设计一个 DropDownList2_SelectedIndexChanged 事件处理过程，它的功能就是根据 DropDownList2 中的值确定 DropDownList3 的可选项。这里需要将 DropDownList2 的 AutoPostBack 属性设为 True，以便在用户选择一个地区项后进行回发，执行上述事件处理过程。但出现一个问题，

DropDownList2 中默认显示第一个地区项，只有用户选择其他地区项时才引发执行 DropDownList2_SelectedIndexChanged，为此，在 DropDownList2 的绑定数据集中插入一个空选项，用户只能选择非空地区项(因为它对应有一个非空验证控件)，从而保证执行该事件处理过程。

图 12.27 Registered 的设计界面

对于省份下拉列表 DropDownList3 和市下拉列表 DropDownList4 采用同样的设计。

用户注册的所有数据项都是必输项，而且要满足相关要求，如邮箱格式正确，年龄在 10～99 之间等。在用户成功输入后，单击“确定”按钮会将该游客信息添加到 Customers 表中。

本网页对应的 C#代码如下：

```
using System;
using System.Web.UI;
using System.Data;
public partial class Customer_Registered : System.Web.UI.Page
{   CommDB mydb = new CommDB();                          //创建 CommDB 类对象
    DataSet myds = new DataSet();
    string mysql;                                        //存放 SQL 语句
    protected void Page_Load(object sender, EventArgs e)
    {   if (!Page.IsPostBack)                            //在网页首发时执行
        {   mysql = "SELECT 学历 FROM Education";
            myds = mydb.ExecuteQuery(mysql, "Education");  //执行 SELECT 语句
            DataRow nrow = myds.Tables["Education"].NewRow(); //插入一个空行
            nrow["学历"] = "";
            myds.Tables["Education"].Rows.InsertAt(nrow, 0);
            DropDownList1.DataSource = myds.Tables["Education"];
                                //设置学历下拉列表的数据源
            DropDownList1.DataTextField = "学历";          //设置学历下拉列表的绑定字段
            DropDownList1.DataBind();                    //数据绑定
            mysql = "SELECT distinct 地区 FROM Area";
            myds = mydb.ExecuteQuery(mysql, "Area");     //执行 SELECT 语句
```

```
            DataRow nrow1 = myds.Tables["Area"].NewRow();   //插入一个空行
            nrow1["地区"] = "";
            myds.Tables["Area"].Rows.InsertAt(nrow1, 0);
            DropDownList2.DataSource = myds.Tables["Area"];
                            //设置地区下拉列表的数据源
            DropDownList2.DataTextField = "地区";           //设置地区下拉列表的绑定字段
            DropDownList2.DataBind();                       //数据绑定
            DropDownList3.Enabled = false;
            DropDownList4.Enabled = false;
            DropDownList5.Enabled = false;
        }
        else                                                //在回传时保持密码
        {   passTextBox1.Attributes.Add("value", passTextBox1.Text);
            passTextBox2.Attributes.Add("value", passTextBox2.Text);
        }
    }
    protected void DropDownList2_SelectedIndexChanged(object sender, EventArgs e)
    {   DropDownList3.Enabled = true;
        mysql = "SELECT distinct 省份 FROM Area WHERE 地区 = '"
         + DropDownList2.SelectedValue.ToString().Trim() + "'";
        myds = mydb.ExecuteQuery(mysql, "Area");            //执行 SELECT 语句
        DataRow nrow = myds.Tables["Area"].NewRow();        //插入一个空行
        nrow["省份"] = "";
        myds.Tables["Area"].Rows.InsertAt(nrow, 0);
        DropDownList3.DataSource = myds.Tables["Area"];     //设置省份下拉列表的数据源
        DropDownList3.DataTextField = "省份";               //设置省份下拉列表的绑定字段
        DropDownList3.DataBind();                           //数据绑定
        DropDownList4.Items.Clear();
        DropDownList5.Items.Clear();
    }
    protected void DropDownList3_SelectedIndexChanged(object sender, EventArgs e)
    {   DropDownList4.Enabled = true;
        mysql = "SELECT distinct 市 FROM Area WHERE 省份 = '"
         + DropDownList3.SelectedValue.ToString().Trim() + "'";
        myds = mydb.ExecuteQuery(mysql, "Area");            //执行 SELECT 语句
        DataRow nrow = myds.Tables["Area"].NewRow();        //插入一个空行
        nrow["市"] = "";
        myds.Tables["Area"].Rows.InsertAt(nrow, 0);
        DropDownList4.DataSource = myds.Tables["Area"];     //设置市下拉列表的数据源
        DropDownList4.DataTextField = "市";                 //设置市下拉列表的绑定字段
        DropDownList4.DataBind();                           //数据绑定
        DropDownList5.Items.Clear();
    }
    protected void DropDownList4_SelectedIndexChanged(object sender, EventArgs e)
    {   DropDownList5.Enabled = true;
        mysql = "SELECT distinct 县 FROM Area WHERE 市 = '"
         + DropDownList4.SelectedValue.ToString().Trim() + "'";
        myds = mydb.ExecuteQuery(mysql, "Area");            //执行 SELECT 语句
        DataRow nrow = myds.Tables["Area"].NewRow();
        nrow["县"] = "";
        myds.Tables["Area"].Rows.InsertAt(nrow, 0);
        DropDownList5.DataSource = myds.Tables["Area"];     //设置县下拉列表的数据源
        DropDownList5.DataTextField = "县";                 //设置县下拉列表的绑定字段
        DropDownList5.DataBind();                           //数据绑定
    }
```

```
    protected void Button1_Click(object sender, EventArgs e)
    {    if (Page.IsValid)
         {    int i;
              mysql  =  "SELECT  *  FROM Customers WHERE 用户名 = '"
               + usernameTextBox.Text.Trim()  +  "'";
              i  =  mydb.Rownum(mysql);                       //执行 SQL 语句并返回行数 i
              if (i > 0)                                      //顾客用户名重复
                   Response.Write("<script>alert('对不起,你输入的用户名"
                    + "已经注册了!')</script>");
              else                                            //合法的用户名
              {    mysql  =  "INSERT INTO Customers(用户名,密码,姓名,年龄,学历," +
                    "地区,省份,市,县,住址,邮箱,电话,有效否)"
                    + "VALUES('"  +  usernameTextBox.Text.Trim()  +  "','"
                    + passTextBox1.Text.Trim()  +  "','"
                    + xmTextBox.Text.Trim()  +  "',"
                    + ageTextBox.Text  +  ",'"
                    + DropDownList1.SelectedValue.ToString().Trim()  +  "','"
                    + DropDownList2.SelectedValue.ToString().Trim()  +  "','"
                    + DropDownList3.SelectedValue.ToString().Trim()  +  "','"
                    + DropDownList4.SelectedValue.ToString().Trim()  +  "','"
                    + DropDownList5.SelectedValue.ToString().Trim()  +  "','"
                    + placeTextBox.Text.Trim()  +  "','"
                    + EmailTextBox.Text.Trim()  +  "','"
                    + TelTextBox.Text.Trim()  +  "','1')";
                   mydb.ExecuteNonQuery(mysql);
                   Response.Redirect("~/dispinfo.aspx?info = 你可以退出再以顾客身份"
                       +"登录后购物,或者继续以游客身份浏览!");
              }
         }
    }
}
```

例如,某游客(其用户名为 wh10)的注册如图 12.28 所示,单击"确定"按钮即完成注册。



图 12.28　Registered 网页的执行界面

注意：对于本网页中的 5 个下拉列表，初始时游客不能单击“省份”、“市”和“县”下拉列表。只有单击“地区”下拉列表并选中一个非空地区项后，“省份”下拉列表才可用，而“市”和“县”下拉列表仍不可用；只有单击“省份”下拉列表并选中一个非空省份项后，“市”下拉列表才可用，而“县”下拉列表仍不可用；只有单击“市”下拉列表并选中一个非空市项后，“县”下拉列表才可用。

12.6.3 “查看(浏览)商品”功能网页设计

查看(浏览)商品由 QueryProduct 和 QueryProduct1 两个网页实现。

1. QueryProduct 网页设计

QueryProduct 网页用于设置查找商品的条件设置，其设计界面如图 12.29 所示。



图 12.29　QueryProduct 网页的设计界面

QueryProduct 网页的设计原理与 Registered.aspx 网页类似。用户在分类下拉列表选择一个项后，在子类下拉列表列出所有该分类的子类项；用户在子类下拉列表选择一个项后，在品牌下拉列表列出所有该子类的品牌项。

最后将用户的输入和执行构成一个条件表达式 condstr，再产生 SELECT 语句 mysql，将其存储在 Session["sql"]中，并通过执行 Response.Redirect("QueryProduct1.aspx")语句转向 QueryProduct1 网页。

说明：这里没有处理用户的 SQL 注入攻击，可以采用第 11 章介绍的方法解决 SQL 注入攻击问题。

本网页对应的 C#代码如下：

```
using System;
using System.Web.UI;
using System.Data;
public partial class QueryProduct : System.Web.UI.Page
{   CommDB mydb = new CommDB();                          //创建 CommDB 类对象
    DataSet myds = new DataSet();
    string mysql;                                        //SELECT 语句
    protected void Page_Load(object sender, EventArgs e)
    {   if (!Page.IsPostBack)                            //在网页首发时执行
        {   mysql = "SELECT distinct 分类 FROM ProdType";
            myds = mydb.ExecuteQuery(mysql, "ProdType");   //执行 SELECT 语句
            DataRow nrow = myds.Tables["ProdType"].NewRow(); //插入一个空行
            nrow["分类"] = "";
```

```
            myds.Tables["ProdType"].Rows.InsertAt(nrow, 0);
            DropDownList1.DataSource = myds.Tables["ProdType"];
                //设置分类下拉列表的数据源
            DropDownList1.DataTextField = "分类";           //设置分类下拉列表的绑定字段
            DropDownList1.DataBind();                       //数据绑定
            DropDownList2.Enabled = false;
            DropDownList3.Enabled = false;
        }
    }
    protected void Button1_Click(object sender, EventArgs e)
    {   string condstr = "有效否 = 1";                      //只查找有效的商品
        //以下构造查询表达式 condstr
        if (bhTextBox.Text != "")
            condstr = "商品编号 Like '" + bhTextBox.Text.Trim() + "%'";
        if (DropDownList1.SelectedValue.ToString().Trim() != "")
            condstr += " AND 分类 = '"
             + DropDownList1.SelectedValue.ToString().Trim() + "'";
        if (DropDownList2.SelectedValue.ToString().Trim() != "")
            condstr += " AND 子类 = '"
             + DropDownList2.SelectedValue.ToString().Trim() + "'";
        if (DropDownList3.SelectedValue.ToString().Trim() != "")
            condstr += " AND 品牌 = '"
             + DropDownList3.SelectedValue.ToString().Trim() + "'";
        float p1 = 0, p2 = 0;
        if (priceTextBox1.Text.Trim() != "")
            p1 = float.Parse(priceTextBox1.Text.Trim());
        if (priceTextBox2.Text.Trim() != "")
            p2 = float.Parse(priceTextBox2.Text.Trim());
        if (p1 != 0.0)
        {   if (p1 <= p2)
                condstr += " AND 单价>=" + p1.ToString().Trim()
                 + " AND 单价<=" + p2.ToString().Trim();
            else
            {   Label1.Text = "错误提示:单价段输入错误.";
                return;
            }
        }
        mysql = "SELECT * FROM Products WHERE " + condstr + " ORDER BY 商品编号";
        Session["sql"] = mysql;                            //用会话保存 SELECT 语句
        Response.Redirect("QueryProduct1.aspx");           //转向 QueryProduct1 网页
    }
    protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
    {   DropDownList2.Enabled = true;
        mysql = "SELECT distinct 子类 FROM ProdType WHERE 分类 = '"
         + DropDownList1.SelectedValue.ToString().Trim() + "'";
        myds = mydb.ExecuteQuery(mysql, "ProdType");       //执行 SELECT 语句
        DataRow nrow = myds.Tables["ProdType"].NewRow();   //插入一个空行
        nrow["子类"] = "";
        myds.Tables["ProdType"].Rows.InsertAt(nrow, 0);
        DropDownList2.DataSource = myds.Tables["ProdType"];
                //设置子类下拉列表的数据源
        DropDownList2.DataTextField = "子类";              //设置子类下拉列表的绑定字段
        DropDownList2.DataBind();                          //数据绑定
        DropDownList3.Items.Clear();
    }
    protected void DropDownList2_SelectedIndexChanged(object sender, EventArgs e)
```

```
        {   DropDownList3.Enabled = true;
            mysql = "SELECT distinct 品牌 FROM ProdType WHERE 子类 = '"
              + DropDownList2.SelectedValue.ToString().Trim() + "'";
            myds = mydb.ExecuteQuery(mysql, "ProdType");        //执行 SELECT 语句
            DataRow nrow = myds.Tables["ProdType"].NewRow();    //插入一个空行
            nrow["品牌"] = "";
            myds.Tables["ProdType"].Rows.InsertAt(nrow, 0);
            DropDownList3.DataSource = myds.Tables["ProdType"];
                    //设置品牌下拉列表的数据源
            DropDownList3.DataTextField = "品牌";                //设置品牌下拉列表的绑定字段
            DropDownList3.DataBind();                           //数据绑定
        }
    }
```

2. QueryProduct1 网页设计

QueryProduct1 网页用于显示满足条件的商品信息，包含 WHERE 条件的 SELECT 语句保存在当前会话 Session["sql"]中。QueryProduct1 网页的设计界面如图 12.30 所示。

图 12.30　QueryProduct1 网页的设计界面

其中主要包含一个 GridView1 控件和一个 Button1 控件。GridView1 控件的源视图代码如下：

```
<asp:GridView ID = "GridView1" runat = "server" AllowPaging = "True"
    AutoGenerateColumns = "False" BackColor = "LightGoldenrodYellow"
    BorderColor = "Tan" BorderWidth = "1px" CellPadding = "2"
    Font - Bold = "True" Font - Size = "10pt" ForeColor = "Black" GridLines = "None"
    Width = "680px" OnPageIndexChanging = "GridView1_PageIndexChanging" PageSize = "5" >
    <FooterStyle BackColor = "Tan" />
    <Columns>
      <asp:BoundField DataField = "商品编号" HeaderText = "商品编号">
        <HeaderStyle Font - Bold = "True" Font - Size = "18px" Font - Names = "隶书"
          ForeColor = "Blue" />
        <ItemStyle HorizontalAlign = "Center" />
      </asp:BoundField>
      <asp:BoundField DataField = "分类" HeaderText = "分类">
```

```
        <HeaderStyle Font-Bold="True" Font-Size="18px" Font-Names="隶书"
          ForeColor="Blue" />
        <ItemStyle HorizontalAlign="Center" />
      </asp:BoundField>
      <asp:BoundField DataField="子类" HeaderText="子类">
        <HeaderStyle Font-Bold="True" Font-Size="18px" Font-Names="隶书"
          ForeColor="Blue" />
        <ItemStyle HorizontalAlign="Center" />
      </asp:BoundField>
      <asp:BoundField DataField="品牌" HeaderText="品牌" >
        <HeaderStyle Font-Bold="True" Font-Size="18px" Font-Names="隶书"
          ForeColor="Blue" />
        <ItemStyle HorizontalAlign="Center" />
      </asp:BoundField>
      <asp:BoundField DataField="型号" HeaderText="型号" >
        <HeaderStyle Font-Bold="True" Font-Size="18px" Font-Names="隶书"
          ForeColor="Blue" />
        <ItemStyle HorizontalAlign="Center" />
      </asp:BoundField>
      <asp:BoundField DataField="单价" HeaderText="单价" >
        <HeaderStyle Font-Bold="True" Font-Size="18px" Font-Names="隶书"
          ForeColor="Blue" />
        <ItemStyle HorizontalAlign="Center" />
      </asp:BoundField>
      <asp:TemplateField HeaderText="图片">
        <EditItemTemplate>
          <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        </EditItemTemplate>
        <ItemTemplate>
          <asp:Image ID="Image1" runat="server" Height="50px" Width="50px"
            ImageUrl='<%# DataBinder.Eval(Container.DataItem,
              "图片").ToString().Trim() %>'/>
        </ItemTemplate>
        <HeaderStyle Font-Bold="True" Font-Names="隶书" Font-Size="18px"
          ForeColor="Blue" />
         <ItemStyle HorizontalAlign="Center" />
      </asp:TemplateField>
    </Columns>
    <SelectedRowStyle BackColor="DarkSlateBlue" ForeColor="GhostWhite" />
    <PagerStyle BackColor="PaleGoldenrod" ForeColor="DarkSlateBlue"
        HorizontalAlign="Center" />
    <HeaderStyle BackColor="Tan" Font-Bold="True" />
    <AlternatingRowStyle BackColor="PaleGoldenrod" />
</asp:GridView>
```

从上述代码中可以看出 GridView1 控件的设计思路。设计难点是图片字段，GridView1 控件对应的数据源中图片字段存放的是图像文件，这里需要显示该图像，所以将图片字段转换为 TemplateField，进入编辑模板对话框，删除 ItemTemplate 中原来内容，添加一个 Image 控件 Image1，其属性设置如图 12.31 所示。

本网页对应的 C#代码如下：

```
using System;
using System.Data;
public partial class QueryProduct1 : System.Web.UI.Page
```

```
{   string mysql;                                           //SQL 表达式
    CommDB mydb = new CommDB();                             //创建 CommDB 类对象
    DataSet myds = new DataSet();                           //创建 DataSet 对象
    protected void Page_Load(object sender, EventArgs e)
    {   if (!Page.IsPostBack)
        {
            bind();                                         //在网页首发时执行
        }
    }
    public void bind()
    {   //自定义方法
        mysql = Session["sql"].ToString();                  //获取用会话保存的 SELECT 语句
        myds = mydb.ExecuteQuery(mysql, "Products");
        GridView1.DataSource = myds.Tables["Products"];
        GridView1.DataKeyNames = new string[] { "商品编号" };
        GridView1.DataBind();                     //在 GridView1 控件中显示满足查询条件的记录
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        Response.Redirect("～/dispinfo.aspx?info=欢迎使用本系统!");
    }
    protected void GridView1_PageIndexChanging(object sender,
       System.Web.UI.WebControls.GridViewPageEventArgs e)
    {   //分页
        GridView1.PageIndex = e.NewPageIndex;
        bind();                                             //在分页时调用自定义方法
    }
}
```



图 12.31　编辑图片字段的模板

例如，某游客进入本网站后，在查找商品中选择“电脑办公”分类下的“笔记本”子类，如图 12.32 所示。单击“确定”按钮，出现如图 12.33 所示的商品显示界面。此时，游客只能查看，不能进行其他操作。



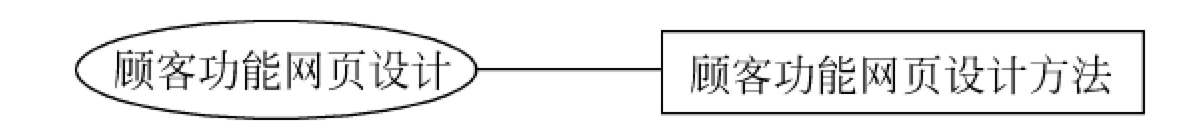
图 12.32 QueryProduct 网页的执行界面



图 12.33 QueryProduct1 网页的执行界面

12.7 顾客功能网页设计

知识梳理



12.7.1 顾客功能主页设计

当顾客进入网站后，首先显示顾客功能主页 customermenu.aspx，其设计界面如图 12.34 所示。它是基于母版页 MasterPage.master 的。该网页位于网站根目录下，它所调用的所有

网页都放置在子 Customer 目录中。



图 12.34 顾客主页 customermenu. 的设计界面

customermenu.aspx 网页设计与游客功能主页 touristmenu.aspx 相似，只是 TreeView1 控件是节点和节点的链接网页不同而已。TreeView1 控件的源视图代码如下：

```
<asp:TreeView ID="TreeView1" runat="server" style="text-align:center;
    font-family:仿宋;font-weight:bold;font-size: 16px">
    <Nodes>
      <asp:TreeNode Text="购物管理" Value="购物管理"
        NavigateUrl="dispinfo.aspx?info=欢迎使用本系统" Target="Iframe1">
        <asp:TreeNode Text="选购商品放入购物车" Value="选购商品放入购物车"
          NavigateUrl="~/Customer/ShoppingCart.aspx" Target="Iframe1" />
        <asp:TreeNode Text="编辑我的购物车" Value="编辑我的购物车"
          NavigateUrl="~/Customer/editShoppingCart.aspx" Target="Iframe1" />
        <asp:TreeNode Text="购物车结算" Value="购物车结算"
          NavigateUrl="~/Customer/buyproduct.aspx" Target="Iframe1" />
      </asp:TreeNode>
      <asp:TreeNode Text="我的订单管理" Value="我的订单管理"
        NavigateUrl="dispinfo.aspx?info=欢迎使用本系统" Target="Iframe1">
        <asp:TreeNode Text="查看我的订单" Value="查看我的订单"
         NavigateUrl="~/Customer/dispmyod.aspx" Target="Iframe1"></asp:TreeNode>
        <asp:TreeNode Text="撤销尚未处理的订单" Value="撤销尚未处理的订单"
          NavigateUrl="~/Customer/backrollod.aspx" Target="Iframe1"/>
        <asp:TreeNode Text="订单商品评价" Value="订单商品评价"
          NavigateUrl="~/Customer/productevaluation.aspx" Target="Iframe1" />
      </asp:TreeNode>
      <asp:TreeNode Text="我的信息管理" Value="我的信息管理"
        NavigateUrl="dispinfo.aspx?info=欢迎使用本系统" Target="Iframe1">
        <asp:TreeNode Text="更改我的信息" Value="更改我的信息"
         NavigateUrl="~/Customer/updatecustomerinfo.aspx" Target="Iframe1" />
        <asp:TreeNode Text="更改我的密码" Value="更改我的密码"
         NavigateUrl="~/Customer/updatecustomerpass.aspx" Target="Iframe1" />
```

```
        </asp:TreeNode>
    </Nodes>
</asp:TreeView>
```

本网页对应的 C#代码如下：

```
using System;
public partial class customermenu : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Text = "顾客端→欢迎你:" + Session["uname"];
    }
}
```

例如，前面注册的 wh10 顾客进入本网站后的执行界面如图 12.35 所示。用户可以单击 TreeView1 控件的节点执行相应的功能。



图 12.35　顾客主页 customermenu 的执行界面

12.7.2　“选购商品放入购物车”功能网页设计

该功能由 ShoppingCart 和 ShoppingCart1 两个网页实现。设计原理与 12.6.3 小节介绍的功能网页设计相似。

1. ShoppingCart 网页

ShoppingCart 网页用于设置查找要购买商品的条件，其设计界面如图 12.36 所示。除了表格中第 1 行的 HTML 文字不同外，其他与 ShoppingCart 网页相同。它将用户的输入和执行构成一个条件表达式 condstr，再产生 SELECT 语句 mysql，将其存储在 Session["sql"]中，并通过执行 Response.Redirect("ShoppingCart1.aspx")语句转向 ShoppingCart1 网页。

图 12.36　ShoppingCart 网页的设计界面

2. ShoppingCart1 网页

ShoppingCart1 网页用于显示满足条件的商品信息，并提供用户选购商品和查看商品评价的功能，其设计界面如图 12.37 所示。

图 12.37　ShoppingCart1 网页的设计界面

其中主要包含一个 GridView1 控件和两个命令按钮(Button1 和 Button2)。GridView1 控件的源视图代码如下：

```
<asp:GridView ID = "GridView1" runat = "server" AutoGenerateColumns = "False"
    BackColor = "LightGoldenrodYellow" BorderColor = "Tan" BorderWidth = "1px"
    CellPadding = "2" Font - Bold = "True" Font - Size = "10pt"
    ForeColor = "Black" GridLines = "None" Width = "795px"
    DataKeyNames = "图片" OnRowEditing = "GridView1_RowEditing">
    <FooterStyle BackColor = "Tan" />
    <Columns>
      <asp:BoundField DataField = "商品编号" HeaderText = "商品编号">
        <HeaderStyle Font - Bold = "True" Font - Size = "18px" Font - Names = "隶书"
          ForeColor = "Blue" />
```

```
  <ItemStyle HorizontalAlign = "Center" />
</asp:BoundField>
<asp:BoundField DataField = "分类" HeaderText = "分类">
  <HeaderStyle Font - Bold = "True" Font - Size = "18px" Font - Names = "隶书"
    ForeColor = "Blue" />
  <ItemStyle HorizontalAlign = "Center" />
</asp:BoundField>
<asp:BoundField DataField = "子类" HeaderText = "子类">
  <HeaderStyle Font - Bold = "True" Font - Size = "18px" Font - Names = "隶书"
    ForeColor = "Blue" />
  <ItemStyle HorizontalAlign = "Center" />
</asp:BoundField>
<asp:BoundField DataField = "品牌" HeaderText = "品牌" >
  <HeaderStyle Font - Bold = "True" Font - Size = "18px" Font - Names = "隶书"
     ForeColor = "Blue" />
  <ItemStyle HorizontalAlign = "Center" />
</asp:BoundField>
<asp:BoundField DataField = "型号" HeaderText = "型号" >
  <HeaderStyle Font - Bold = "True" Font - Size = "18px" Font - Names = "隶书"
    ForeColor = "Blue" />
  <ItemStyle HorizontalAlign = "Center" />
</asp:BoundField>
<asp:BoundField DataField = "单价" HeaderText = "单价" >
   <HeaderStyle Font - Bold = "True" Font - Size = "18px" Font - Names = "隶书"
     ForeColor = "Blue" />
   <ItemStyle HorizontalAlign = "Center" />
</asp:BoundField>
<asp:TemplateField HeaderText = "图片">
  <ItemTemplate>
   <asp:Image ID = "Image1" runat = "server" Height = "50px" Width = "50px"
      ImageUrl = '<% # DataBinder.Eval(Container.DataItem,"图片").
                ToString().Trim() %>' />
  </ItemTemplate>
  <HeaderStyle Font - Bold = "True" Font - Names = "隶书" Font - Size = "18px"
     ForeColor = "Blue" />
  <ItemStyle HorizontalAlign = "Center" />
</asp:TemplateField>
<asp:BoundField DataField = "星数" HeaderText = "星数" >
  <HeaderStyle Font - Bold = "True" Font - Size = "18px" Font - Names = "隶书"
    ForeColor = "Blue" />
  <ItemStyle HorizontalAlign = "Center" />
</asp:BoundField>
<asp:BoundField DataField = "库存数量" HeaderText = "库存数量" >
   <HeaderStyle Font - Bold = "True" Font - Size = "18px" Font - Names = "隶书"
     ForeColor = "Blue" />
   <ItemStyle HorizontalAlign = "Center" />
</asp:BoundField>
<asp:TemplateField HeaderText = "放入否" ShowHeader = "False">
  <HeaderStyle Font - Bold = "True" Font - Size = "18px" Font - Names = "隶书"
     ForeColor = "Blue" />
  <ItemStyle HorizontalAlign = "Center" />
  <ItemTemplate>
    <asp:CheckBox ID = "CheckBox1" runat = "server" />
  </ItemTemplate>
</asp:TemplateField>
```

```
        <asp:TemplateField HeaderText = "数量">
          <HeaderStyle Font - Bold = "True" Font - Size = "18px" Font - Names = "隶书"
             ForeColor = "Blue" />
          <ItemStyle HorizontalAlign = "Center" />
          <ItemTemplate>
            <asp:TextBox ID = "TextBox1" runat = "server" Height = "16px"
              Width = "24px"> 0 </asp:TextBox>
          </ItemTemplate>
        </asp:TemplateField>
        <asp:CommandField ButtonType = "Button" EditText = "查看评价"
           ShowEditButton = "True" >
           <ControlStyle Font - Bold = "True" ForeColor = "Red" />
        </asp:CommandField>
      </Columns>
      <SelectedRowStyle BackColor = "DarkSlateBlue" ForeColor = "GhostWhite" />
      <PagerStyle BackColor = "PaleGoldenrod" ForeColor = "DarkSlateBlue"
        HorizontalAlign = "Center" />
      <HeaderStyle BackColor = "Tan" Font - Bold = "True" />
      <AlternatingRowStyle BackColor = "PaleGoldenrod" />
  </asp:GridView>
```

GridView1 控件关联 Products 商品表，其中“放入否”和“数量”并没有绑定的表字段，它们都是 TemplateField 字段，“放入否”字段的 ItemTemplate 模板中放置一个复选框 CheckBox1，“数量”字段的 ItemTemplate 模板中放置一个文本框 TextBox1。“查看评价”为 CommandField 字段，只有 ShowEditButton 属性为 True，在用户单击时引发执行 GridView1_RowEditing 事件处理方法。

本网页对应的 C# 代码如下：

```
using System;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
public partial class ShoppingCart1 : System.Web.UI.Page
{   string mysql;                                       //SQL 表达式
    CommDB mydb = new CommDB();                         //创建 CommDB 类对象
    DataSet myds = new DataSet();                       //创建 DataSet 对象
    protected void Page_Load(object sender, EventArgs e)
    {   if (!Page.IsPostBack)
        {   bind();                                     //在网页首发时执行
            if (GridView1.Rows.Count == 0)
            {   Label1.Text = "没有任何满足条件的商品信息!";
                Button1.Visible = false;
            }
            else
                Button2.Visible = false;
        }
    }
    public void bind()
    {                                                   //自定义方法
        mysql = Session["sql"].ToString();              //获取用会话保存的 SELECT 语句
        myds = mydb.ExecuteQuery(mysql, "Products");
        GridView1.DataSource = myds.Tables["Products"];
        GridView1.DataKeyNames = new string[] { "商品编号" };
        GridView1.DataBind();                   //在 GridView1 控件中显示满足查询条件的记录
```

```
}
protected void Button1_Click(object sender, EventArgs e)
{   if (checkerror())                               //顾客输入出现错误
    {   Label1.Text = "错误提示：你选择的商品数量不正确,重新输入";
        return;
    }
    savedata();                                     //保存到购物车中
    Response.Redirect("~/dispinfo.aspx?info = 你选择的商品已保存到购物车中!");
}
protected bool checkerror()                     //检查数量输入错误的情况,出错返回 true
{   CheckBox xzBox;                                 //复选框对象
    TextBox slBox;                                  //文本框对象
    int i, sl,kcsl;
    for (i = 0; i < GridView1.Rows.Count; i++)
    {   xzBox = GridView1.Rows[i].FindControl("CheckBox1") as CheckBox;
            //在该行中找 CheckBox1 控件
        slBox = GridView1.Rows[i].FindControl("TextBox1") as TextBox;
            //在该行中找 TextBox1 控件
        if (xzBox.Checked)
        {   sl = int.Parse(slBox.Text.Trim());      //提取该行输入的数量
            kcsl = int.Parse(GridView1.Rows[i].Cells[8].Text.Trim());
                //提取该行商品的库存数量
            if (sl <= 0)                        //如果勾选了复选框,而文本框输入 0 时出错
                return true;
            if (sl > kcsl)                          //购买数量大于库存数量
                return true;
        }
    }
    return false;
}
protected void savedata()                           //自定义过程,保存到购物车中
{   string spno;                                    //商品编号
    CheckBox xzBox;                                 //复选框对象
    TextBox slBox;                                  //文本框对象
    Image imgBox;                                   //图片框对象
    int i;
    for (i = 0; i < GridView1.Rows.Count; i++)      //扫描 GridView1 中所有行
    {   xzBox = GridView1.Rows[i].FindControl("CheckBox1") as CheckBox;
        slBox = GridView1.Rows[i].FindControl("TextBox1") as TextBox;
        if (xzBox.Checked)                          //若选择了该行的商品
        {   spno = GridView1.Rows[i].Cells[0].Text.Trim();
                //提取该行的商品编号
            imgBox = GridView1.Rows[i].FindControl("Image1") as Image;
            if (inCart(spno))                       //所选择的商品是在购物车中
            {   mysql = "UPDATE ShoppingCart SET 数量 = 数量 + "
                  + slBox.Text.Trim()
                  + " WHERE 用户名 = '" + Session["uname"]
                  + "' AND 商品编号 = '" + spno + "'";
            }
            else
            {   string f1 = GridView1.Rows[i].Cells[1].Text.Trim();
                    //提取该行的商品分类
                string f2 = GridView1.Rows[i].Cells[2].Text.Trim();
                    //提取该行的商品子类
                string f3 = GridView1.Rows[i].Cells[3].Text.Trim();
```

```
                    //提取该行的商品品牌
                string f4 = GridView1.Rows[i].Cells[4].Text.Trim();
                    //提取该行的商品型号
                string f5 = GridView1.Rows[i].Cells[5].Text.Trim();
                    //提取该行的商品单价
                string f6 = imgBox.ImageUrl;            //提取该行的商品图片
                string f7 = slBox.Text.Trim();          //提取该行的数量
                mysql = "INSERT INTO ShoppingCart(用户名,商品编号,分类,"
                  + "子类,品牌,型号,单价,图片,数量) VALUES('"
                  + Session["uname"] + "','"
                  + spno + "','" + f1 + "','" + f2 + "','"
                  + f3 + "','" + f4 + "','" + f5 + "','"
                  + f6 + "'," + f7 + ")";
            }
            mydb.ExecuteNonQuery(mysql);                //执行 SQL 语句
            mysql = "UPDATE ShoppingCart SET 金额 = 数量 * 单价"
             + " WHERE 用户名 = '" + Session["uname"]
             + "' AND 商品编号 = '" + spno + "'";
            mydb.ExecuteNonQuery(mysql);                //执行 SQL 语句
        }
    }
}
protected bool inCart(string spno)                      //判断所选择的商品是否在购物车中
{   int i;
    mysql = "SELECT * FROM ShoppingCart WHERE 用户名 = '"
     + Session["uname"] + "' AND 商品编号 = '" + spno + "'";
    i = mydb.Rownum(mysql);
    if (i > 0)
        return true;
    else
        return false;
}
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{   e.Cancel = true;
    string spbh = GridView1.DataKeys[e.NewEditIndex][0].ToString().Trim();
    Response.Redirect("dispcomment.aspx?spbh = " + spbh);
}
protected void Button2_Click(object sender, EventArgs e)
{
    Response.Redirect("~/dispinfo.aspx?info = 欢迎使用本系统!");
}
}
```

当用户执行 ShoppingCart1 网页并单击“确定”按钮返回时，需要对用户操作进行如下处理：

- 若选中某个商品记录，而购买数量为 0，这是不允许的，通过自定义方法 checkerror 来检查。
- 若用户的购物车中已经存在本次又购买的商品，则需要将购物车中该商品数量增加，这是通过自定义方法 inCart 实现的。
- 如果用户操作正确，将本次购物存储到购物车中，这是通过自定义方法 savedata 实现的。

例如，wh10 顾客的一次购物操作如图 12.38 所示，单击“确定”按钮后，将购物信息存放到购物车中，购物车中对应的数据如图 12.39 所示。



选购商品放入购物车

商品编号	分类	子类	品牌	型号	单价	图片	星数	库存数量	放入否	数量	
2111	电脑办公	笔记本	戴尔	XPS13R-9343-2508	7999		4	71	☐	0	查看评价
2112	电脑办公	笔记本	戴尔	XPS13R-9343-1708	8500		0	39	☑	1	查看评价
2121	电脑办公	笔记本	ThinkPad	X250-20CLA06-BCD	5689		0	81	☐	0	查看评价
2122	电脑办公	笔记本	ThinkPad	E450-20DCA01-HCD	4159		0	80	☑	2	查看评价

确　定

图 12.38　ShoppingCart1 网页的执行界面

结果　消息

	用户名	商品编号	分类	子类	品牌	型号	图片	单价	数量	金额
1	wh10	2112	电脑办公	笔记本	戴尔	XPS13R-9343-1708	~//Picture//2112.jpg	8500	1	8500
2	wh10	2122	电脑办公	笔记本	ThinkPad	E450-20DCA01-HCD	~//Picture//2122.jpg	4159	2	8318

图 12.39　wh10 顾客对应的购物车数据

12.7.3　“编辑我的购物车”功能网页设计

编辑我的购物车网页为 editShoppingCart，这里顾客只能更新所购商品的数量，如果放弃某个商品，可以将对应的数量设置为 0。

editShoppingCart 网页的设计界面如图 12.40 所示，其中 GridView1 控件和相关事件处理方法设计思路与 ShoppingCart1 网页相似。



图 12.40　editShoppingCart 网页的设计界面

在顾客更新购物车并单击“保存购物车”按钮 Button1 后，Button1_Click 事件处理方法执行过程是：先从购物车表 ShoppingCart 中删除该顾客(Session["uname"])的所有信息；然后

循环处理 GridView1 控件中的每一行,如果该行商品的购物数量大于 0,则将该行信息重新插入到 ShoppingCart 表中。

例如,wh10 顾客进入编辑我的购物车,将 2122 编号的商品购买数量修改为 1,如图 12.41 所示,单击“保存购物车”按钮。



图 12.41 editShoppingCart 网页的执行界面

12.7.4 “购物车结算”功能网页设计

购物车结算功能由 buyproduct 和 Orderform 两个网页来完成。

说明: 顾客购物车中保存顾客临时购物信息,只要不结算,该信息总是有效。顾客可以多次购物操作,但购物车只要一个,一旦结算,购物车信息会转入到 OrderForm 订单表中,并将购物车清空,一次结算产生一个订单号,订单号是系统根据 OrderForm 表中的订单数量自动生成的。

1. buyproduct 网页设计

buyproduct 网页的设计界面如图 12.42 所示,GridView1 控件中显示当前顾客在购物车中存放的所有商品信息,顾客只能查看,不能编辑。其设计与 ShoppingCart1 网页中的 GridView1 控件相似。



图 12.42 buyproduct 网页的设计界面

本网页中“确定结算”按钮 Button1 对应的事件处理方法如下：

```
protected void Button1_Click(object sender, EventArgs e)
{   int i;
    //(1)在 Products 表中更新各商品的库存数量
    for (i = 0; i < GridView1.Rows.Count; i++)
    {   string spno = GridView1.Rows[i].Cells[0].Text.Trim();          //提取该行的商品编号
        string gwsl = GridView1.Rows[i].Cells[7].Text.Trim();
                //提取该行的商品购物数量
        Label1.Text = "数量:" + gwsl;
        mysql = "UPDATE Products SET 库存数量 = 库存数量 - " + gwsl
          + " WHERE 商品编号 = '" + spno + "'";
        mydb.ExecuteNonQuery(mysql);                         //执行 SQL 语句
    }
    //(2)通过 Sales 表求本订单的订单号
    mysql = "SELECT COUNT( * ) FROM "
     + "(SELECT distinct 订单号 FROM Sales) tmp";
    string dds = mydb.ExecuteAggregateQuery(mysql);          //求订单数
    string ndds = (int.Parse(dds) + 1).ToString();           //新订单编号
    Session["ndds"] = ndds;
    //(3)将订单的顾客信息插入到 OrderForm 表中
    string name,dq,sf,cs, xm, dz, yx, th;
    mysql = "SELECT 姓名,地区,省份,市,县,住址,邮箱,电话 FROM Customers" +
     " WHERE 用户名 = '" + Session["uname"] + "'";
    myds = mydb.ExecuteQuery(mysql, "Customers");
    DataRow mydr = myds.Tables["Customers"].Rows[0];
        //获取查询结果集的第 1 行(查询结果集只有 1 行)
    name = mydr["姓名"].ToString().Trim();
    dq = mydr["地区"].ToString().Trim();
    sf = mydr["省份"].ToString().Trim();
    cs = mydr["市"].ToString().Trim();
    xm = mydr["县"].ToString().Trim();
    dz = mydr["住址"].ToString().Trim();
    yx = mydr["邮箱"].ToString().Trim();
    th = mydr["电话"].ToString().Trim();
    Session["name"] = name;                                  //收件人姓名
    Session["sjrdz"] = sf + cs + xm + dz;                    //收件人地址
    Session["th"] = th;                                      //收件人电话
    mysql = "INSERT INTO OrderForm(订单号,日期,用户名,姓名,地区,"
     + "省份,市,县,住址,邮箱,电话,总数量,总金额,处理否,结算否) VALUES("
     + ndds + ",'" + DateTime.Now + "','" + Session["uname"] + "','"
     + name + "','" + dq + "','" + sf + "','" + cs + "','"
     + xm + "','" + dz + "','" + yx + "','" + th + "',"
     + Session["zsl"] + "," + Session["zjr"] + ",0,0)";
    Label1.Text = mysql;
    mydb.ExecuteNonQuery(mysql);                             //执行 SQL 语句更新数据库
    //(4)将购物车全部信息作为订单移到 Sales 表中
    mysql = "INSERT INTO Sales (订单号,日期,用户名,"
     + "商品编号,分类,子类,品牌,型号,单价,数量,金额) "
     + "SELECT "
     + ndds + ",'"
     + DateTime.Now + "',"
     + "sc.用户名,sc.商品编号,sc.分类,sc.子类,"
     + "sc.品牌,sc.型号,sc.单价,sc.数量,sc.金额 "
     + "FROM ShoppingCart sc "
     + "WHERE 用户名 = '" + Session["uname"] + "'";
    mydb.ExecuteNonQuery(mysql);                             //执行 SQL 语句更新数据库
    mysql = "DELETE ShoppingCart"                            //删除原来的购物车记录
```

```
        + " WHERE 用户名 = '" + Session["uname"] + "'";
    mydb.ExecuteNonQuery(mysql);                        //执行 SQL 语句更新数据库
    Response.Redirect("Orderform.aspx");                //转向 Orderform 网页
}
```

2. Orderform 网页设计

Orderform 网页十分简单，设计界面如图 12.43 所示，包含一个多行文本框 TextBox1 和一个 Button1，通过如下 Page_Load 事件处理方法显示顾客订单信息：

```
protected void Page_Load(object sender, EventArgs e)
{   string ndds = Session["ndds"].ToString().Trim();
    string zsl = Session["zsl"].ToString().Trim();
    string zjr = Session["zjr"].ToString().Trim();
    string name = Session["name"].ToString().Trim();        //收件人姓名
    string sjrdz = Session["sjrdz"].ToString().Trim();      //收件人地址
    string th = Session["th"].ToString().Trim();            //收件人电话
    TextBox1.Text = "订单信息\r\n"
     + "订单编号：" + ndds +"\r\n"
     + "购买商品：一共购买" + zsl + "件商品，总金额为" + zjr + "元\r\n"
     + "收 件 人：" + name + "\r\n"
     + "送货地址：" + sjrdz + "\r\n"
     + "你的电话：" + th;
}
```



图 12.43　OrderForm 网页的设计界面

例如，wh10 顾客进入购物车结算，如图 12.44 所示，单击“确定结算”按钮，出现如图 12.45 所示的订单显示界面，顾客单击“确定”按钮返回。



购物车-结算

商品编号	分类	子类	品牌	型号	单价	图片	数量	金额
2112	电脑办公	笔记本	戴尔	XPS13R-9343-1708	8500		1	8500
2122	电脑办公	笔记本	ThinkPad	E450-20DCA01-HCD	4159		1	4159

订单：你一共购买2件商品，总金额为12659元

确定结算　暂时放弃结算

图 12.44　buyproduct 网页的执行界面

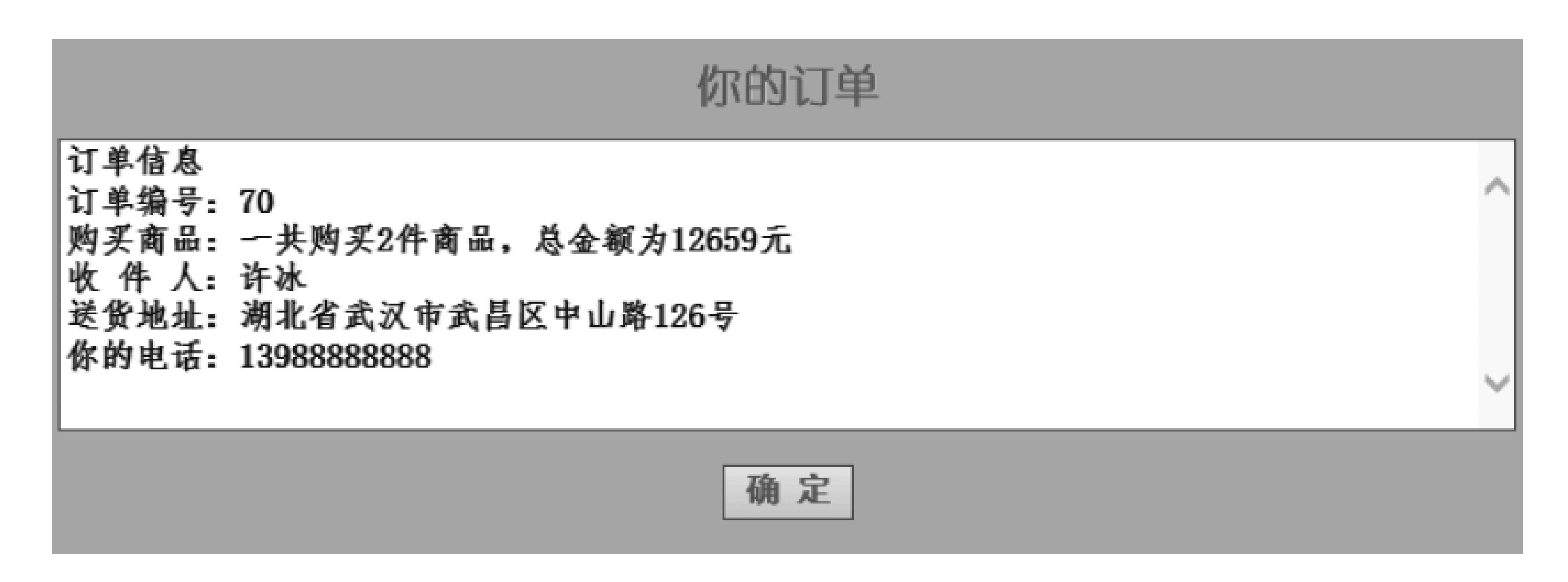


图 12.45　OrderForm 网页的执行界面

12.7.5　“查看我的订单”功能网页设计

查看我的订单功能由 dispmyod 和 dispmyod1 两个网页实现。

1. dispmyod 网页设计

dispmyod 网页的设计界面如图 12.46 所示。

dispmyod.aspx

查看我的订单

订单号	日期	商品总数量	总金额	处理否	
数据绑定	数据绑定	数据绑定	数据绑定	□	查看订单商品
数据绑定	数据绑定	数据绑定	数据绑定	□	查看订单商品
数据绑定	数据绑定	数据绑定	数据绑定	□	查看订单商品
数据绑定	数据绑定	数据绑定	数据绑定	□	查看订单商品
数据绑定	数据绑定	数据绑定	数据绑定	□	查看订单商品

确 认

图 12.46　dispmyod 网页的设计界面

GridView1 控件中的“处理否”字段表示操作员是否已经处理了该订单。一旦处理，该订单不能撤销。该字段是一个 TemplateField 类型，其 ItemTemplate 设计模板如图 12.47 所示，绑定的 trans 方法如下：

```
public bool trans(object s)
{   if (s.ToString().Trim() == "True")
        return true;
    else
        return false;
}
```

GridView1 控件中的“查看订单商品”字段是 CommandField 类型的字段，其源视图代码如下：

```
<asp:CommandField ButtonType = "Button" EditText = "查看订单商品"
    ShowEditButton = "True">
    <ControlStyle Font - Bold = "True" ForeColor = "Red" />
    <ItemStyle HorizontalAlign = "Center" />
</asp:CommandField>
```

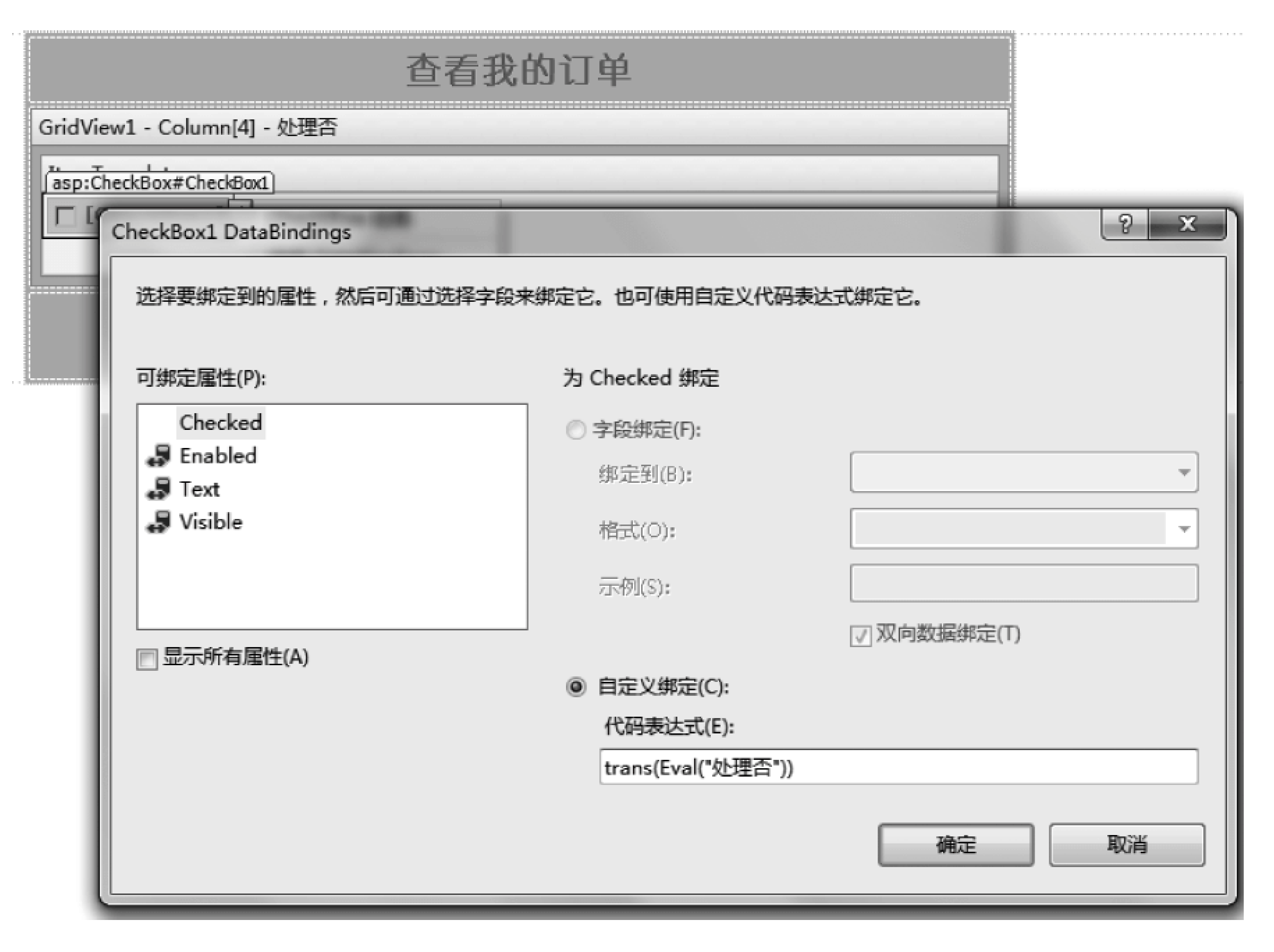


图 12.47 “处理否”字段设计

在用户单击某行的“查看订单商品”按钮时引发执行的事件处理方法如下：

```
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{   e.Cancel = true;
    string ddh = GridView1.DataKeys[e.NewEditIndex][0].ToString();
    Session["ddh"] = ddh;
    Response.Redirect("dispmyod1.aspx");
}
```

在提取订单 ddh 并保存在 Session["ddh"]中之后转向 dispmyod1 网页。

2. dispmyod1 网页设计

dispmyod 网页的设计界面如图 12.48 所示。该网页用于显示指定订单号的商品信息，这是通过如下 Page_Load 事件处理方法实现的：



图 12.48 dispmyod1 网页的设计界面

```
protected void Page_Load(object sender, EventArgs e)
{   if (!Page.IsPostBack)
    {   mysql = "SELECT 日期,商品编号,分类,子类,品牌,型号,单价,数量,金额"
          + " FROM Sales"
          + " WHERE 用户名 = '" + Session["uname"] + "'"
          + " AND 订单号 = '" + Session["ddh"] + "'";
        myds = mydb.ExecuteQuery(mysql, "Sales");
        GridView1.DataSource = myds.Tables["Sales"];
        GridView1.DataKeyNames = new string[] { "商品编号" };
        GridView1.DataBind();                    //在 GridView1 控件中显示满足查询条件的记录
    }
}
```

例如，wh10 顾客进入查看我的订单，显示结果如图 12.49 所示，表明该顾客有一个订单，订单编号为 70，操作员尚未处理。在该行单击“查看订单商品”按钮，显示结果如图 12.50 所示，列出该订单对应的所有商品信息，单击“返回”按钮完成本次查看操作。



图 12.49　dispmyod 网页的执行界面

你所选订单包含的商品信息

日期	商品编号	分类	子类	品牌	型号	单价	数量	金额
2015/8/1 0:00:00	2112	电脑办公	笔记本	戴尔	XPS13R-9343-1708	8500	1	8500
2015/8/1 0:00:00	2122	电脑办公	笔记本	ThinkPad	E450-20DCA01-HCD	4159	1	4159

返 回

图 12.50　dispmyod1 网页的执行界面

12.7.6　“撤销尚未处理的订单”功能网页设计

撤销尚未处理的订单功能对应的网页为 backrollod，其设计界面如图 12.51 所示。该网页主要包含一个 GridView1 控件。

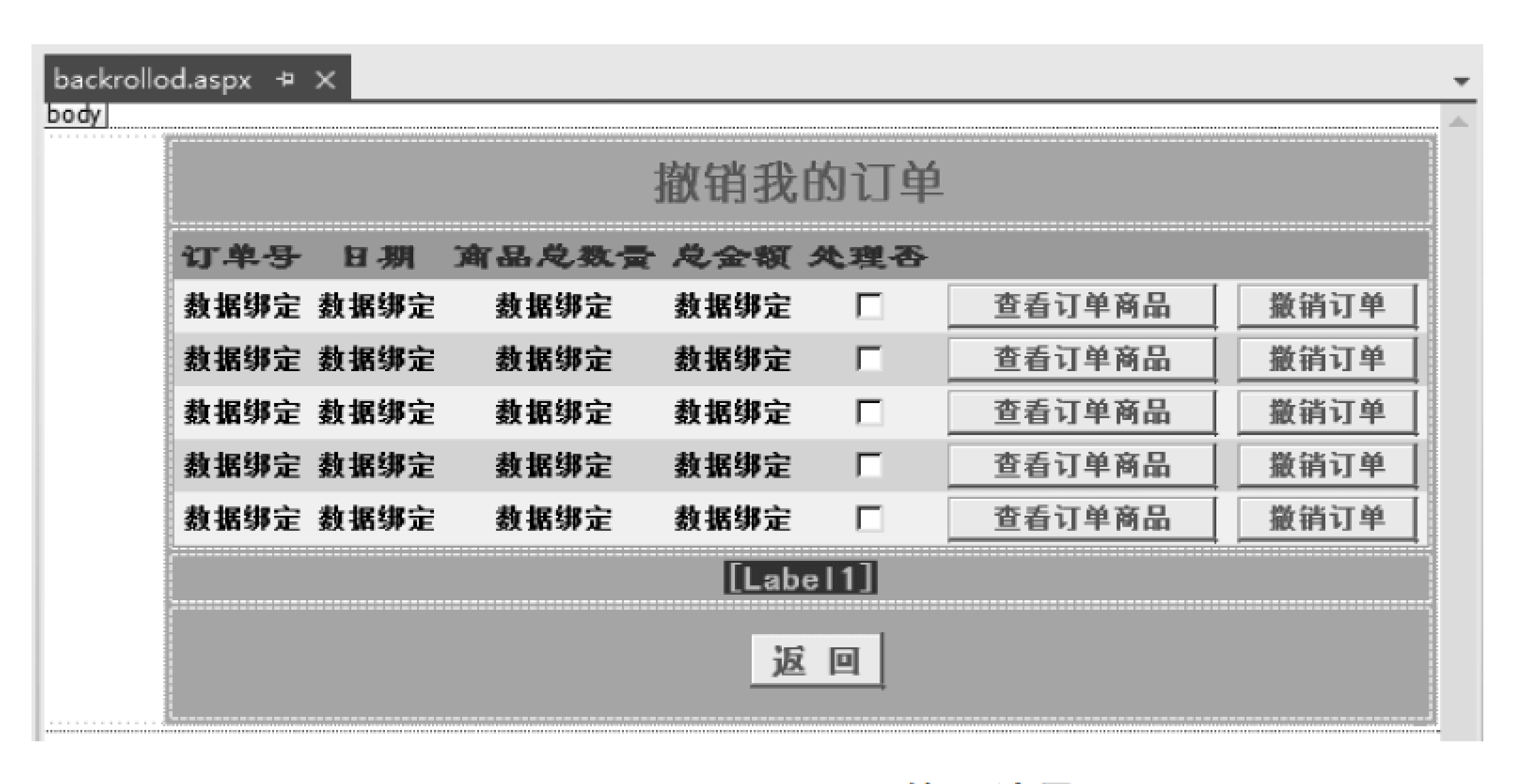


图 12.51　dispmyod1 网页的设计界面

GridView1 控件的设计与 dispmyod 网页在 GridView1 控件设计相似，只增加了一个 CommandField 字段，其源视图代码如下：

```
<asp:CommandField ButtonType = "Button" ShowSelectButton = "True"
    SelectText = "撤销订单">
    <ControlStyle Font - Bold = "True" ForeColor = "Red" />
    <ItemStyle HorizontalAlign = "Center" />
    <HeaderStyle Font - Bold = "True" Font - Size = "18px" Font - Names = "隶书"
      ForeColor = "Blue" />
</asp:CommandField>
```

该字段绑定的 GridView1_SelectedIndexChanged 事件处理方法如下：

```
protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
{   CheckBox chkbox;
    chkbox = GridView1.SelectedRow.FindControl("CheckBox1") as CheckBox;
    if (chkbox.Checked)
        Label1.Text = "该订单已处理,不能撤销!";
    else                                                        //撤销订单
    {   //(1)对于 Sales 表中该订单的每种商品,修改 Products 表中该商品的库存数量
        string spno,sl;
        string ddh = GridView1.SelectedRow.Cells[0].Text.ToString().Trim();
            //提取所选行的订单号
        mysql = "SELECT * FROM Sales" + " WHERE 订单号 = '" + ddh + "'";
        myds = mydb.ExecuteQuery(mysql, "Sales");
        DataTable dt = myds.Tables["Sales"];
        DataRow[] myRow = dt.Select();
        foreach (DataRow row in myRow)                          //逐行处理数据
        {   spno = row["商品编号"].ToString().Trim();
            sl = row["数量"].ToString().Trim();
            mysql = "UPDATE Products SET 库存数量 = 库存数量 + " + sl
              + " WHERE 商品编号 = '" + spno + "'";
            mydb.ExecuteNonQuery(mysql);                        //执行 SQL 语句
        }
        //(2)删除 Sales 表中该订单的所有记录
        mysql = "DELETE Sales" + " WHERE 订单号 = '" + ddh + "'";
        mydb.ExecuteNonQuery(mysql);
        //(3)删除 OrderForm 表中该订单的记录
        mysql = "DELETE OrderForm" + " WHERE 订单号 = '" + ddh + "'";
        mydb.ExecuteNonQuery(mysql);
        Label1.Text = ddh + "订单已撤销";
    }
}
```

上述事件处理方法用于撤销所在行的订单，其过程分为 3 步，详细实现见代码中的注释。

12.7.7 “订单商品评价”功能网页设计

订单商品评价功能由 productevaluation 和 writecomment 两个网页实现。

1. productevaluation 网页设计

productevaluation 网页显示该顾客所有订单的全部商品，供顾客评价。其设计界面如

图 12.52 所示。



图 12.52　productevaluation 网页的设计界面

GridView1 控件中的“评价”按钮对应的事件处理方法如下：

```
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{    e.Cancel = true;
     string spbh = GridView1.DataKeys[e.NewEditIndex][0].ToString();
     Session["spbh"] = spbh;
     Response.Redirect("writecomment.aspx");
}
```

通过 Session["spbh"]记录下要评价的商品编号，并转向 writecomment 网页。

2. writecomment 网页设计

writecomment 网页用于给 Session["spbh"]商品打分和写评语，其设计界面如图 12.53 所示。打分采用 5 分制，最高 5 分。



图 12.53　writecomment 网页的设计界面

首先查看该顾客身份是否评价过，如果已经评价过，则显示原来的评价结果；否则，置空评价界面。该过程由 Page_Load 事件处理方法完成。

当顾客单击“提交”按钮后，执行的事件处理方法如下：

```
protected void Button1_Click(object sender, EventArgs e)
{    string fs = "";
     if (Radio1.Checked == true)
         fs = "1";
     else if (Radio2.Checked == true)
         fs = "2";
```

```
        else if (Radio3.Checked == true)
            fs = "3";
        else if (Radio4.Checked == true)
            fs = "4";
        else if (Radio5.Checked == true)
            fs = "5";
        else
            fs = "5";
        if (Session["has"].ToString().Trim() == "0")
        {   mysql = "INSERT INTO Comment(商品编号,用户名,评语,分数) VALUES('"
              + Session["spbh"].ToString().Trim() + "','"
              + Session["uname"].ToString().Trim() + "','"
              + TextBox1.Text + "',"
              + fs + ")";
            mydb.ExecuteNonQuery(mysql);
            mysql = "UPDATE Products SET 评论数 = 评论数 + 1"
              + " WHERE 商品编号 = '" + Session["spbh"].ToString().Trim() + "'";
            mydb.ExecuteNonQuery(mysql);
            mysql = "UPDATE Products SET 星数 = (星数 * (评论数 - 1) + " + fs
              + ")/评论数"
              + " WHERE 商品编号 = '" + Session["spbh"].ToString().Trim() + "'";
            mydb.ExecuteNonQuery(mysql);
        }
        else
        {   mysql = "UPDATE Comment SET "
              + "评语 = '" + TextBox1.Text + "',"
              + "分数 = " + fs
              + " WHERE 用户名 = '"
              + Session["uname"].ToString().Trim() + "' AND 商品编号 = '"
              + Session["spbh"].ToString().Trim() + "'";
            mydb.ExecuteNonQuery(mysql);
            mysql = "UPDATE Products SET 星数 = "
              + "(星数 * 评论数 + " + fs + " - " + Session["fs"].ToString().Trim()
              + ")/评论数"
              + " WHERE 商品编号 = '" + Session["spbh"].ToString().Trim() + "'";
            mydb.ExecuteNonQuery(mysql);
        }
        Response.Redirect("productevaluation.aspx");
    }
```

例如，wh10 顾客进入订单商品评价功能，如图 12.54 所示，显示该顾客购买的两个商品，订单号为 70。单击第 2 行商品的“评价”按钮，转向 writecomment 网页，顾客可以打分和写评语，如该顾客给 2122 商品的评价结果如图 12.55 所示。单击“提交”按钮返回到 productevaluation 网页，可以继续评价其他商品。

订单商品评价

订单号	日期	商品编号	分类	子类	品牌	型号	单价	数量	金额	
70	2015/8/1 0:00:00	2112	电脑办公	笔记本	戴尔	XPS13R-9343-1708	8500	1	8500	评价
70	2015/8/1 0:00:00	2122	电脑办公	笔记本	ThinkPad	E450-20DCA01-HCD	4159	1	4159	评价

返 回

图 12.54　productevaluation 网页的执行界面

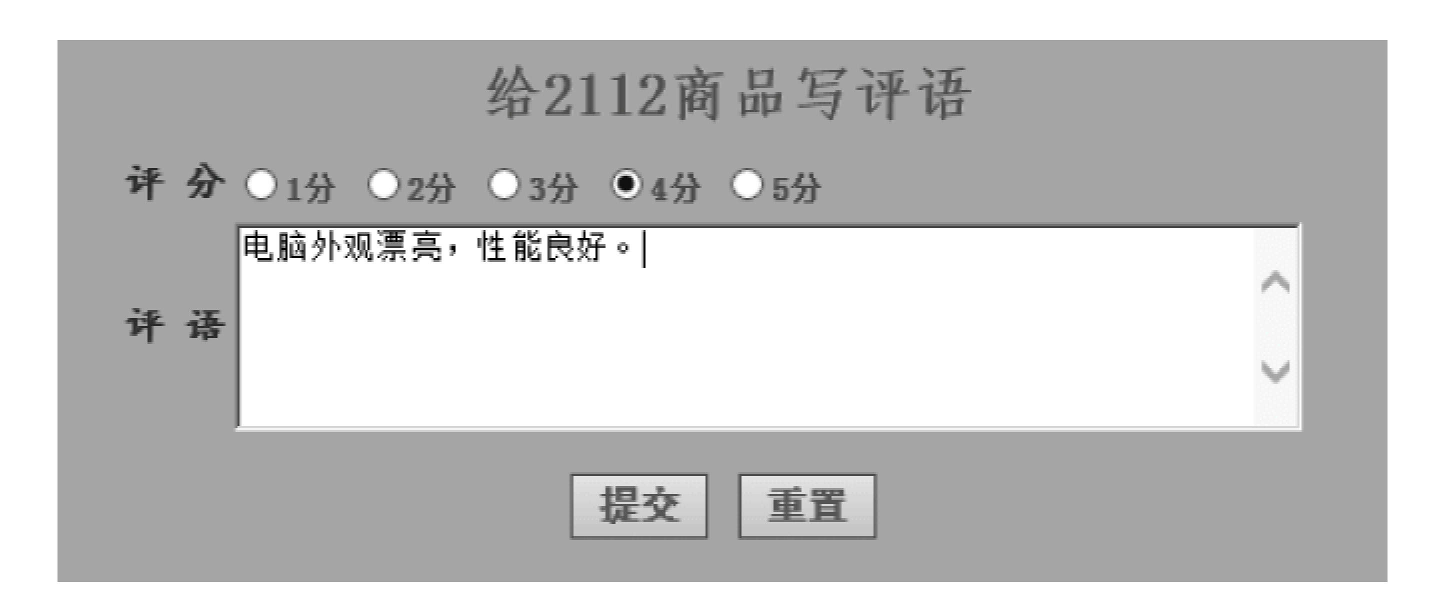


图 12.55　writecomment 网页的执行界面

顾客评价后，在 Products 商品表中会累积所有顾客的评分，从而产生出该商品的星数，计算公式如下：

某商品星数 = 所有顾客的评分/顾客人数

但需要考虑顾客重复评分情况。商品的评语存储在 Comment 表中。

12.7.8　“更改我的信息”功能网页设计

更改我的信息功能对应 updatecustomerinfo 网页，其设计界面如图 12.56 所示。

updatecustomerinfo.aspx
修改我的信息
用户名* 必须输入用户名
真实姓名 必须输入姓名
年 龄* 必须输入年龄 年龄输入错误
学 历* 未绑定 [Label1] 必须选择学历
地 址 地区* 未绑定 必须选择地区
省份* 未绑定 必须选择省份
市* 未绑定 必须选择城市
县* 未绑定 必须选择县
住址* 必须输入住址
邮 箱* 必须输入邮箱 邮箱格式错误
电 话* 必须输入电话
确定 重置

图 12.56　updatecustomerinfo 网页的设计界面

updatecustomerinfo 网页的设计思路与 Registered 网页相似。首先通过 Page_Load 事件处理方法在 Customers 表中找到该顾客(Session["uname"])的注册信息，并显示。用户可以更新该信息，单击“确定”按钮后，对应 Button1_Click 事件处理方法使用 UPDARE 语句修改 Customers 表中该顾客的信息。

设计界面上的验证控件用于验证顾客数据输入的有效性。

例如，wh10 顾客进入更改我的信息功能，显示界面如图 12.57 所示，可以修改各项数据，但一定要满足注册时的相关要求，单击“确定”按钮进行数据更新。



图 12.57　updatecustomerinfo 网页的执行界面

12.7.9　“更改我的密码”功能网页设计

更改我的密码功能对应的网页为 updatecustomerpass，其设计界面如图 12.58 所示。



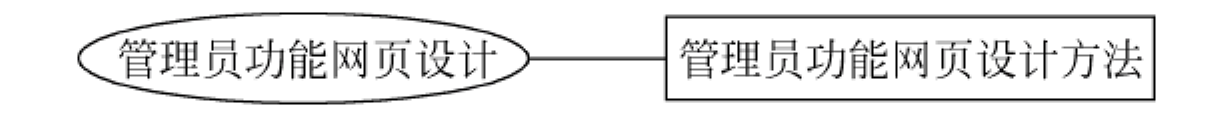
图 12.58　updatecustomerpass 网页的执行界面

当顾客单击“提交”按钮后，调用 Button1_Click 事件处理方法，先检查原密码是否正确，当正确时，用 UPDATE 语句修改该顾客的密码为新密码。

设计界面上的验证控件用于验证顾客数据输入的有效性。

12.8　管理员功能网页设计

知识梳理



12.8.1　管理员功能主页设计

当管理员进入网站后，首先显示管理员功能主页 managermenu，其设计界面如图 12.59 所示。它是基于母版页 MasterPage.master 的。该网页位于网站根目录下，调用的所有网页

都放置在子 Manager 目录中。



图 12.59　managermenu 网页的设计界面

managermenu 网页的设计思路与游客功能主页 touristmenu 的相似，只是在 TreeView1 控件中列出管理员的功能项。

12.8.2　“添加新用户信息”功能网页设计

添加新用户信息功能对应 adduser 网页，其设计界面如图 12.60 所示。

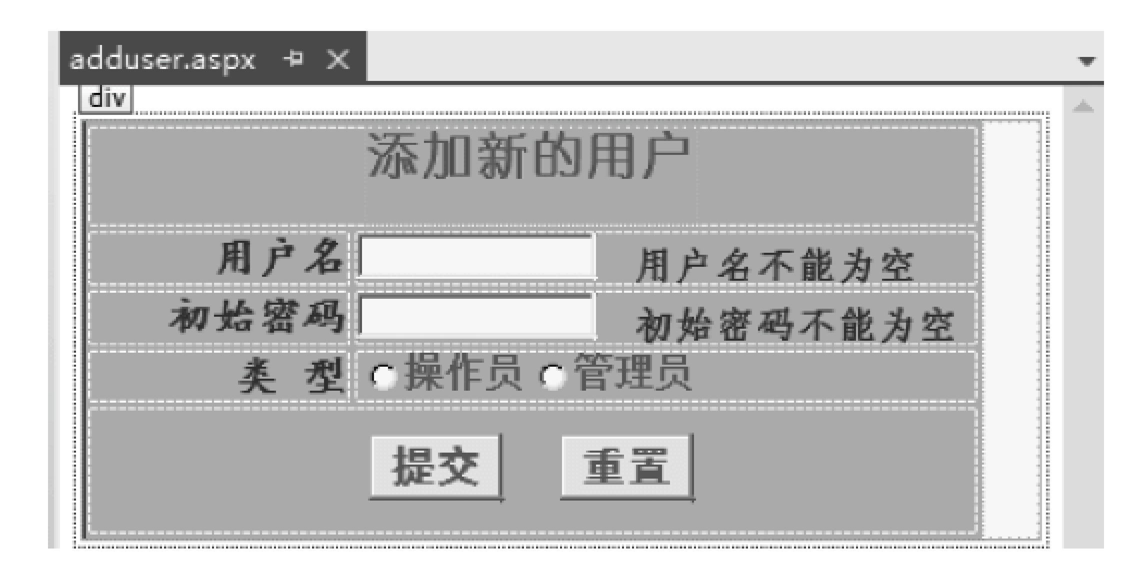


图 12.60　adduser 网页的设计界面

当单击“提交”按钮后，调用 Button1_Click 事件处理方法，首先检查 Users 表中是否重复存在该用户名，若不重复，则使用 INSERT 语句将该用户数据插入到 Users 表中。

设计界面上的验证控件用于验证管理员数据输入的有效性。

12.8.3　“编辑用户信息”功能网页设计

编辑用户信息功能由 edituser、edituser1 和 edituser2 三个网页实现。

1. edituser 网页设计

edituser 网页的设计界面如图 12.61 所示，用于设置查找用户的条件，构成一个条件字符串 condstr，继而生成一个 SELECT 语句，存放在 Session["sql"]会话中，再转向 edituser1 网页。



图 12.61 edituser 网页的设计界面

2. edituser1 网页设计

edituser1 网页的设计界面如图 12.62 所示，主要包含一个 GridView1 控件。

edituser1.aspx

div

编辑用户记录

用户名	类型	有效否	操作1	操作2
数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	编辑	删除
数据绑定	数据绑定	数据绑定	编辑	删除

1 2

退出编辑

图 12.62 edituser1 网页的设计界面

edituser1 网页首先执行 Session["sql"]中的 SELECT 语句，然后将结果在 GridView1 控件中显示出来。

GridView1 控件的两个命令字段如下：

```
<asp:CommandField ShowEditButton = "True" HeaderText = "操作 1" >
    <HeaderStyle Font - Bold = "True" Font - Size = "18px" Font - Names = "隶书"
       ForeColor = "Blue" />
    <ItemStyle ForeColor = "DeepSkyBlue" HorizontalAlign = "Center" />
</asp:CommandField>
<asp:CommandField ShowDeleteButton = "True" HeaderText = "操作 2" >
    <HeaderStyle Font - Bold = "True" Font - Size = "18px" Font - Names = "隶书"
```

```
        ForeColor = "Blue" />
    < ItemStyle ForeColor = "Red" HorizontalAlign = "Center" />
</asp:CommandField >
```

“操作 1”为编辑按钮,单击时引发的事件处理方法如下:

```
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{   //编辑记录
    e.Cancel = true;
    string uname;                                     //保存用户名
    uname = GridView1.DataKeys[e.NewEditIndex].Value.ToString();
    Response.Redirect("edituser2.aspx?uname = " + uname);    //转向 edituser2 网页
}
```

“操作 2”为删除按钮,单击时引发的事件处理方法如下:

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{   //删除记录
    e.Cancel = true;
    string uname;                                     //用户名
    uname = GridView1.DataKeys[e.RowIndex].Value.ToString();
    mysql = "DELETE FROM Users WHERE 用户名 = '" + uname + "'";
    mydb.ExecuteNonQuery(mysql);                      //执行 DELETE 语句
    bind();                                           //调用自定义绑定方法
}
```

3. edituser2 网页设计

edituser2 网页的设计界面如图 12.63 所示,它在管理员单击 edituser1 网页中某个用户行的“编辑”按钮时出现,用于更新该用户的信息。



图 12.63　edituser2 网页的设计界面

12.8.4　“查看顾客信息”功能网页设计

查看顾客信息功能由 dispcustomer 和 dispcustomer1 两个网页实现。

1. dispcustomer 网页设计

dispcustomer 网页的设计界面如图 12.64 所示,它用于设置查找顾客的条件,构成一个条件字符串 condstr,继而生成一个 SELECT 语句,存放在 Session["sql"]会话中,再转向 dispcustomer1 网页。



图 12.64 dispcustomer 网页的设计界面

2. dispcustomer1 网页设计

dispcustomer1 网页的设计界面如图 12.65 所示，主要包含一个 GridView1 控件。它首先执行 Session["sql"]中的 SELECT 语句，然后将结果在 GridView1 控件中显示。GridView1 控件只有分页功能，不能对顾客记录进行编辑和删除操作。

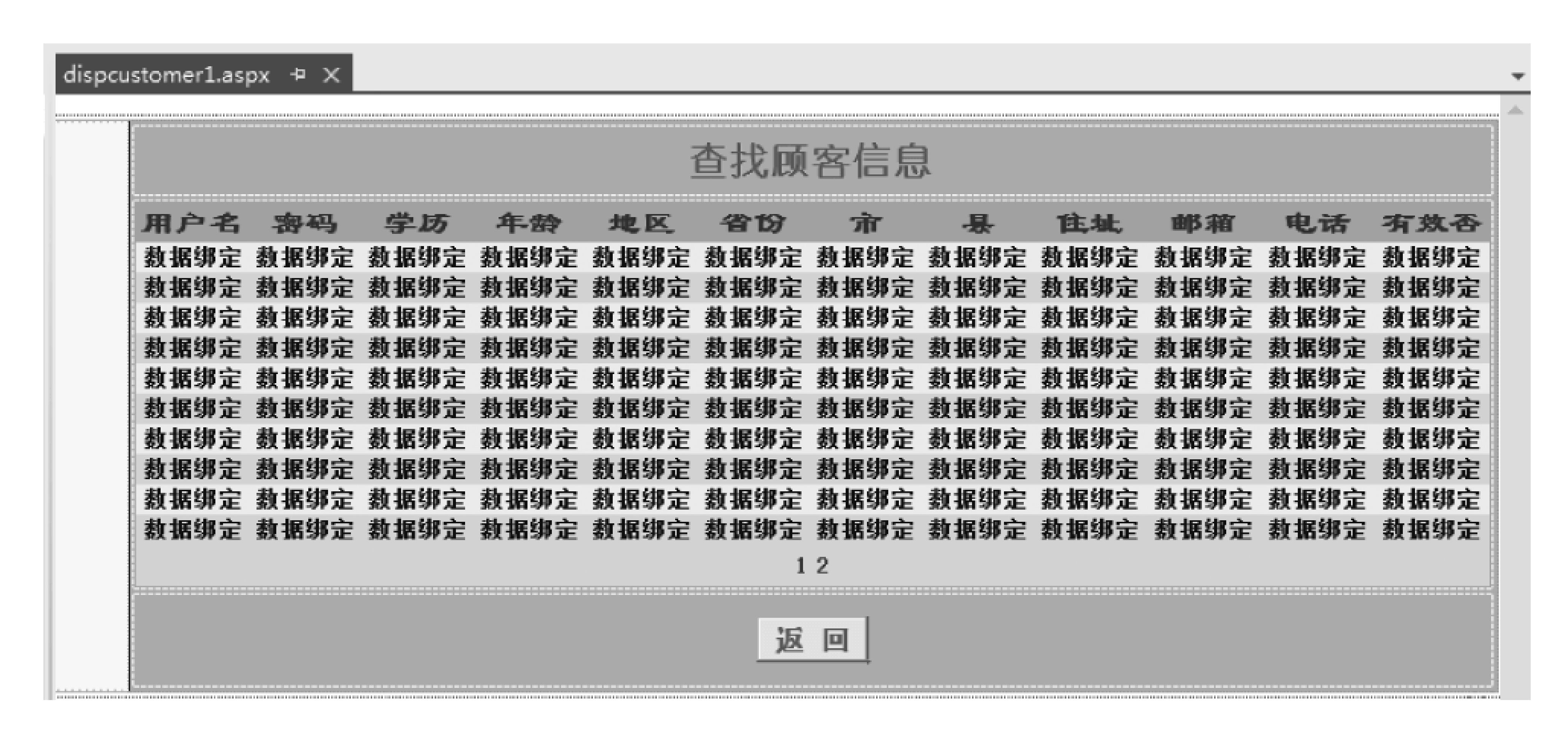


图 12.65 dispcustomer1 网页的设计界面

12.8.5 “临时封杀顾客信息”功能网页设计

临时封杀顾客信息功能由 killcustomer 和 killcustomer1 两个网页实现。

1. killcustomer 网页设计

killcustomer 网页和 dispcustomer 网页相似，其设计界面如图 12.66 所示，它用于设置查找顾客的条件，构成一个条件字符串 condstr，继而生成一个 SELECT 语句，存放在 Session["sql"]会话中，再转向 killcustomer1 网页。

2. killcustomer1 网页设计

killcustomer1 网页的设计界面如图 12.67 所示，主要包含一个 GridView1 控件。它先执

killcustomer.aspx
form#form1
查找临时封杀的顾客信息
用户名
年　龄
学　历 未绑定
地　址 地区 未绑定 省份 未绑定 市 未绑定 县 未绑定
邮　箱
电　话
确定　重置

图 12.66　killcustomer 网页的设计界面



图 12.67　killcustomer1 网页的设计界面

行 Session["sql"]中的 SELECT 语句，然后将结果在 GridView1 控件中显示。

GridView1 控件中有一个“有效否”字段，其源视图代码如下：

```
<asp:TemplateField HeaderText = "有效否">
    <ItemTemplate>
      <asp:CheckBox ID = "CheckBox1" runat = "server"
           Checked = '<% # DataBinder.Eval(Container.DataItem,"有效否") %>' />
    </ItemTemplate>
    <HeaderStyle Font - Bold = "True" Font - Names = "隶书"
       Font - Size = "18px" ForeColor = "Blue" />
    <ItemStyle HorizontalAlign = "Center" />
</asp:TemplateField>
```

它是一个 TemplateField 字段，与 Customers 表“有效否”列绑定。当某顾客有效时，GridView1 控件中会选中该字段，如果管理员取消选中，在单击“确定”按钮后会临时封杀该顾客信息。

例如，管理员 system/manager 进入临时封杀顾客信息，出现 killcustomer 网页执行界面，不选任何项和不输入任何数据，表示查找所有顾客，单击“确定”按钮，出现如图 12.68 所示的界面，管理员可以对任何顾客进行临时封杀操作，单击“确定”按钮后操作变为有效。



hz2	1234	初中及以下	15	华东	浙江省	杭州市	西湖区	G小区123	hz2@123.com	54326789	☑
km1	1234	初中及以下	58	西南	云南省	昆明市	五华区	L小区802	km1@126.com	78978900	☑
nl1	1234	本科	45	华南	广西	南宁市	青秀区	R小区101	nl1@456.com	22222228	☑
nz1	1234	初中及以下	15	西北	甘肃省	兰州市	安宁区	Z小区802	nz1@125.com	54321765	☑
sh1	1234	硕士	35	华东	上海	上海市	黄浦区	C小区888	sh1@126.com	12345678	☑
sh2	1234	专科	23	华东	上海	上海市	徐汇区	D小区901	sh2@126.com	99990999	☑
sjz1	1234	高中	18	华北	河北省	石家庄	长安区	C小区401	sjz1@sina.com	68771234	☑
sjz2	1234	初中及以下	65	华北	河北省	石家庄	新华区	D小区421	sjz2@123.com	65432102	☑
sy1	1234	硕士	46	东北	辽宁省	沈阳市	和平区	S小区680	sy1@126.com	12345678	☑
sy2	1234	本科	32	东北	辽宁省	沈阳市	沈河区	R小区201	sy2@139.com	56789012	☑
wh1	1234	博士	50	华中	湖北省	武汉市	洪山区	D小区432	wh1@126.com	68779999	☑
wh10	1234	硕士	36	华中	湖北省	武汉市	武昌区	中山路126号	wh10@sina.com	13988888888	☑
wh2	1234	本科	25	华中	湖北省	武汉市	武昌区	F小区311	wh2@126.com	87501234	☑
xan1	1234	本科	34	西北	陕西省	西安市	碑林区	D小区602	xan1@543.com	65437890	☑
xan2	1234	本科	32	西北	陕西省	西安市	雁塔区	D小区501	xan2@126.com	98765432	☑

图 12.68　killcustomer1 网页的执行界面

12.8.6　“查看顾客订单信息”功能网页设计

查看顾客订单信息功能由 dispOrderform、dispOrderform1 和 dispOrderform2 三个网页实现。

1. dispOrderform 网页设计

dispOrderform 网页和 dispcustomer 网页相似，其设计界面如图 12.69 所示，它用于设置查找顾客的条件，构成一个条件字符串 condstr，继而生成一个 SELECT 语句，存放在 Session["sql"]会话中，再转向 dispOrderform1 网页。



图 12.69　dispOrderform 网页的设计界面

2. dispOrderform1 网页设计

dispcustomer1 网页的设计界面如图 12.70 所示，主要包含一个 GridView1 控件。

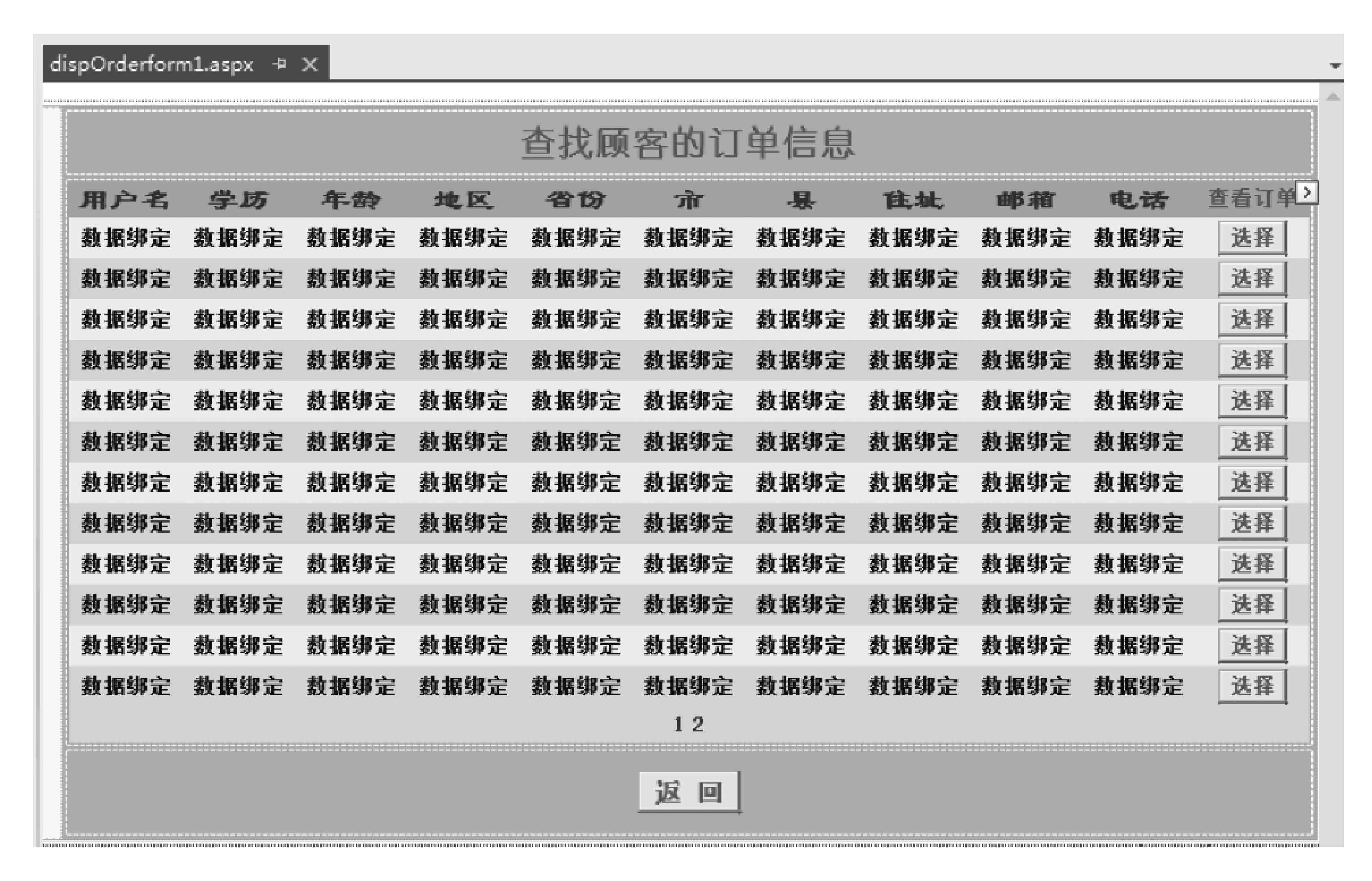


图 12.70　dispOrderform1 网页的设计界面

edituser1 网页首先执行 Session["sql"]中的 SELECT 语句，然后将结果在 GridView1 控件中显示。

GridView1 控件的一个命令字段如下：

```
<asp:CommandField ButtonType = "Button" CancelText = "" DeleteText = "" EditText = "选择"
     HeaderText = "查看订单" ShowEditButton = "True" >
   <ItemStyle HorizontalAlign = "Center" />
   <ControlStyle ForeColor = "Red" Font - Bold = "True" />
   <HeaderStyle ForeColor = "Red" />
   <ItemStyle ForeColor = "Red" />
</asp:CommandField>
```

“选择”为编辑按钮，单击时引发的事件处理方法如下：

```
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{   //编辑记录
    e.Cancel  =  true;
    string cname  =  GridView1.DataKeys[e.NewEditIndex][0].ToString();
          //获取当前行的用户名
    Response.Redirect("dispOrderform2.aspx?cname = "  +  cname);
}
```

3. dispOrderform2 网页设计

dispOrderform2 网页的设计界面如图 12.71 所示。它在管理员单击 dispOrderform1 网页中某个用户行的“选择”按钮时出现，用于显示更新该用户的信息。



图 12.71　dispOrderform2 网页的设计界面

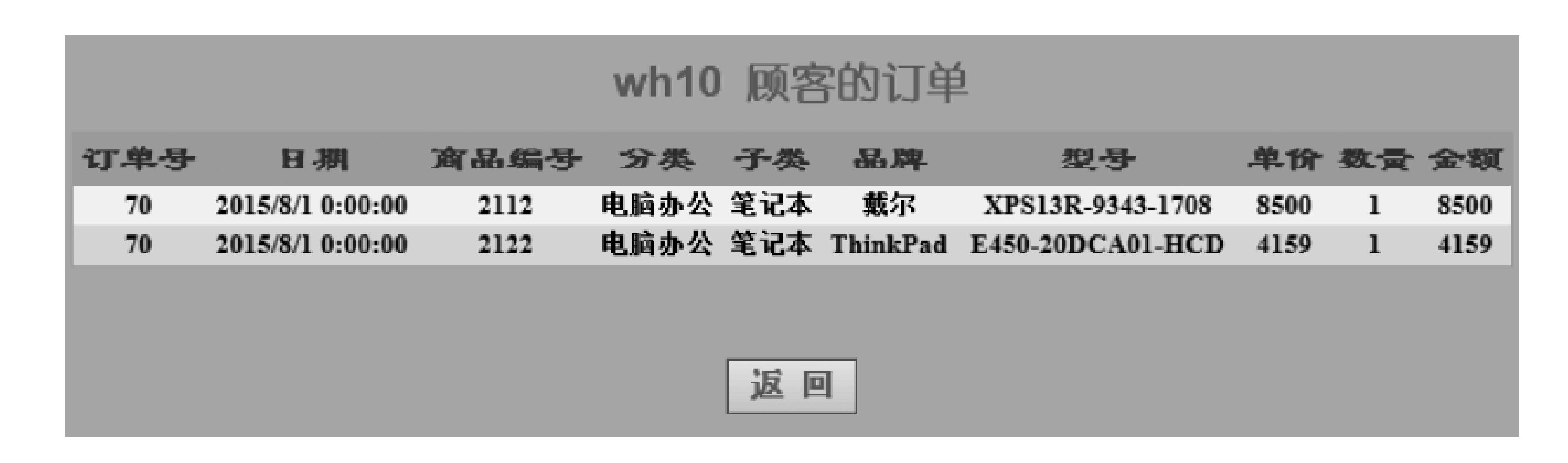
例如，管理员 system/manager 进入查看顾客订单信息，出现 dispOrderform 网页执行界面，选择“华中”地区，单击“确定”按钮，出现如图 12.72 所示的界面，单击 wh10 用户行的“选择”按钮，显示该顾客的所有订单的商品信息，如图 12.73 所示。

查找顾客的订单信息

用户名	学历	年龄	地区	省份	市	县	住址	邮箱	电话	查看订单
cs1	本科	32	华中	湖南省	长沙市	岳麓区	E小区801	cs1@121.com	23456789	选择
wh1	博士	50	华中	湖北省	武汉市	洪山区	D小区432	wh1@126.com	68779999	选择
wh10	硕士	36	华中	湖北省	武汉市	武昌区	中山路126号	wh10@sina.com	13988888888	选择
wh2	本科	25	华中	湖北省	武汉市	武昌区	F小区311	wh2@126.com	87501234	选择

返 回

图 12.72　dispOrderform1 网页的执行界面


wh10 顾客的订单

订单号	日期	商品编号	分类	子类	品牌	型号	单价	数量	金额
70	2015/8/1 0:00:00	2112	电脑办公	笔记本	戴尔	XPS13R-9343-1708	8500	1	8500
70	2015/8/1 0:00:00	2122	电脑办公	笔记本	ThinkPad	E450-20DCA01-HCD	4159	1	4159

返 回

图 12.73　dispOrderform2 网页的执行界面

12.8.7　“商品库存预警”功能网页设计

商品库存预警功能由 Productwarning 和 Productwarning1 两个网页实现。

1. Productwarning 网页设计

Productwarning 网页和 dispcustomer 网页相似，其设计界面如图 12.74 所示，它用于设置查找库存预警商品的条件，构成一个条件字符串 condstr，继而生成一个 SELECT 语句，存放在 Session["sql"]会话中，再转向 Productwarning1 网页。



图 12.74　Productwarning 网页的设计界面

2. Productwarning1 网页设计

Productwarning1 网页的设计界面如图 12.75 所示，主要包含一个 GridView1 控件。它首先执行 Session["sql"]中的 SELECT 语句，然后将结果在 GridView1 控件中显示。GridView1 控件上设计如下事件处理方法：

```
protected void GridView1_RowDataBound(object sender,
        System.Web.UI.WebControls.GridViewRowEventArgs e)
{    if (e.Row.RowIndex >= 0)
     {    int kcsl = int.Parse(e.Row.Cells[6].Text.Trim());    //提取该行的库存数量
          if (kcsl <= 50)
               e.Row.BackColor = System.Drawing.Color.Red;
          else if (kcsl <= 100)
               e.Row.BackColor = System.Drawing.Color.Yellow;
          else
               e.Row.BackColor = System.Drawing.Color.White;
     }
}
```

该方法对不同的库存数量段采用不同的颜色显示。

Productwarning1.aspx

td.auto-style1

商品预警

商品编号	分类	子类	品牌	型号	单价	库存数量
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定

1 2

提示：红色：库存数量<=50；黄色：库存数量<=100；白色为其他

返 回

图 12.75　Productwarning1 网页的设计界面

例如，管理员 system/manager 进入商品库存预警功能，选择所有的商品，出现的库存预警界面如图 12.76 所示。

商品预警

商品编号	分类	子类	品牌	型号	单价	库存数量
1111	手机/数码	手机	小米	红米手机2	749	194
1112	手机/数码	手机	小米	红米note	1067	95
1121	手机/数码	手机	华为	华为P8	2888	82
1122	手机/数码	手机	华为	荣耀6Plus	1999	78
1211	手机/数码	单反数码相机	佳能	EOS70D	7142	42
1212	手机/数码	单反数码相机	佳能	EOS700D	4779	93
1221	手机/数码	单反数码相机	尼康	D7000	4880	94
1222	手机/数码	单反数码相机	尼康	D5300	3580	96
2111	电脑办公	笔记本	戴尔	XPS13R-9343-2508	7999	71
2112	电脑办公	笔记本	戴尔	XPS13R-9343-1708	8500	38
2121	电脑办公	笔记本	ThinkPad	X250-20CLA06-BCD	5689	81
2122	电脑办公	笔记本	ThinkPad	E450-20DCA01-HCD	4159	79

1 2

提示：红色:库存数量<=50；黄色：库存数量<=100；白色为其他

返 回

图 12.76　Productwarning1 网页的执行界面

12.8.8　“商品库存报警”功能网页设计

商品库存报警功能由 Productalarm 和 Productalarm1 两个网页实现。

1．Productalarm 网页设计

Productalarm 网页和 dispcustomer 网页相似，其设计界面如图 12.77 所示。它用于设置查找库存报警商品的条件，构成条件字符串 condstr，继而生成一个 SELECT 语句，存放在 Session["sql"]会话中，再转向 Productalarm1 网页。

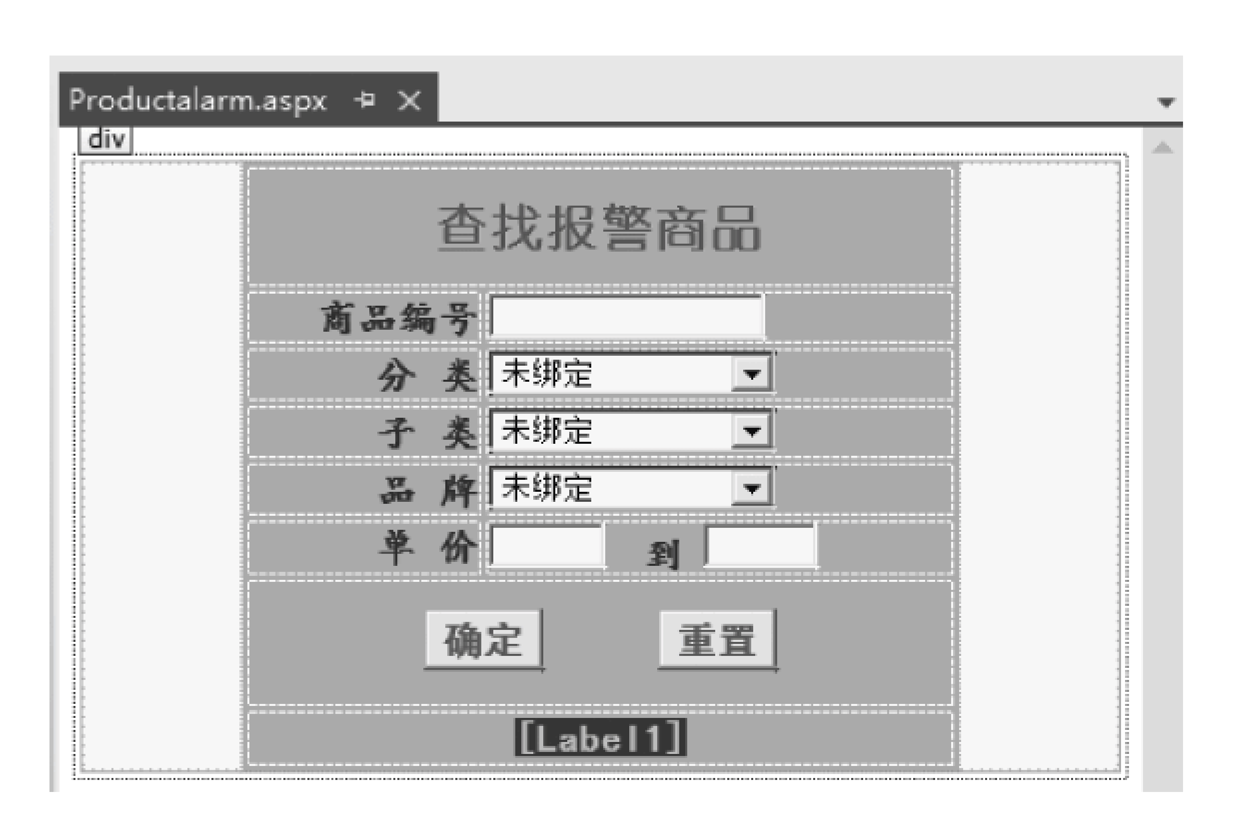


图 12.77　Productalarm 网页的设计界面

2．Productalarm1 网页设计

Productalarm1 网页的设计界面如图 12.78 所示。主要包含一个 GridView1 控件。它首先执行 Session["sql"]中的 SELECT 语句，然后将结果在 GridView1 控件中显示。GridView1 控件上设计如下事件处理方法：

```
protected void GridView1_RowDataBound(object sender,
        System.Web.UI.WebControls.GridViewRowEventArgs e)
{    if (e.Row.RowIndex >= 0)
```

```
    {   string spno = e.Row.Cells[0].Text.Trim();          //提取该行的商品编号
        int kcsl = int.Parse(e.Row.Cells[6].Text.Trim());  //提取该行的库存数量
        mysql = "SELECT COUNT( * ) FROM ShoppingCart "
         + "WHERE 商品编号 = '" + spno + "'";
        int ydsl = int.Parse(mydb.ExecuteAggregateQuery(mysql));
        e.Row.Cells[7].Text = ydsl.ToString().Trim();
        int zr = kcsl - ydsl;
        e.Row.Cells[8].Text = zr.ToString().Trim();
        if (zr <= 0)
            e.Row.BackColor = System.Drawing.Color.Red;
        else if (zr <= 10)
            e.Row.BackColor = System.Drawing.Color.Yellow;
        else
            e.Row.BackColor = System.Drawing.Color.White;
    }
}
```

Productalarm1.aspx

商品报警

商品编号	分类	子类	品牌	型号	单价	库存数量	预订数量	差额
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定		
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定		
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定		
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定		
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定		
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定		
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定		
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定		
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定		
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定		
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定		
数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定	数据绑定		

1 2

提示：红色：差额≤0；黄色：差额≤10；白色为其他

返 回

图 12.78　Productalarm1 网页的设计界面

该方法的思路是，对于某个商品，求出其库存数量和用户购物车中预购该商品的数量，两者之间有一个差额，对不同的差额段采用不同的颜色显示。

例如，管理员 system/manager 进入商品库存报警功能，选择所有的商品，出现的库存报警界面如图 12.79 所示(这里购物车为空)。

商品报警

商品编号	分类	子类	品牌	型号	单价	库存数量	预订数量	差额
1111	手机/数码	手机	小米	红米手机2	749	194	0	194
1112	手机/数码	手机	小米	红米note	1067	95	0	95
1121	手机/数码	手机	华为	华为P8	2888	82	0	82
1122	手机/数码	手机	华为	荣耀6Plus	1999	78	0	78
1211	手机/数码	单反数码相机	佳能	EOS70D	7142	42	0	42
1212	手机/数码	单反数码相机	佳能	EOS700D	4779	93	0	93
1221	手机/数码	单反数码相机	尼康	D7000	4880	94	0	94
1222	手机/数码	单反数码相机	尼康	D5300	3580	96	0	96
2111	电脑办公	笔记本	戴尔	XPS13R-9343-2508	7999	71	0	71
2112	电脑办公	笔记本	戴尔	XPS13R-9343-1708	8500	38	0	38
2121	电脑办公	笔记本	ThinkPad	X250-20CLA06-BCD	5689	81	0	81
2122	电脑办公	笔记本	ThinkPad	E450-20DCA01-HCD	4159	79	0	79

12

提示：红色：差额≤0；黄色：差额≤10；白色为其他

返 回

图 12.79　Productalarm1 网页的执行界面

12.8.9 “商品下架”功能网页设计

商品下架功能由 Productdelete 和 Productdelete1 两个网页实现。

注意：商品下架只是将 Products 表中该商品信息的“有效否”列设置为 0，并不是删除该商品的记录。

1. Productdelete 网页设计

Productdelete 网页和 dispcustomer 网页相似，其设计界面如图 12.80 所示，它用于设置查找下架商品的条件，构成一个条件字符串 condstr，继而生成一个 SELECT 语句，存放在 Session["sql"]会话中，再转向 Productdelete1 网页。



图 12.80 Productdelete 网页的设计界面

2. Productdelete1 网页设计

Productdelete1 网页的设计界面如图 12.81 所示，主要包含一个 GridView1 控件。它首先执行 Session["sql"]中的 SELECT 语句，然后将结果在 GridView1 控件中显示。GridView1 控件中有一个“下架否”字段，其源视图代码如下：

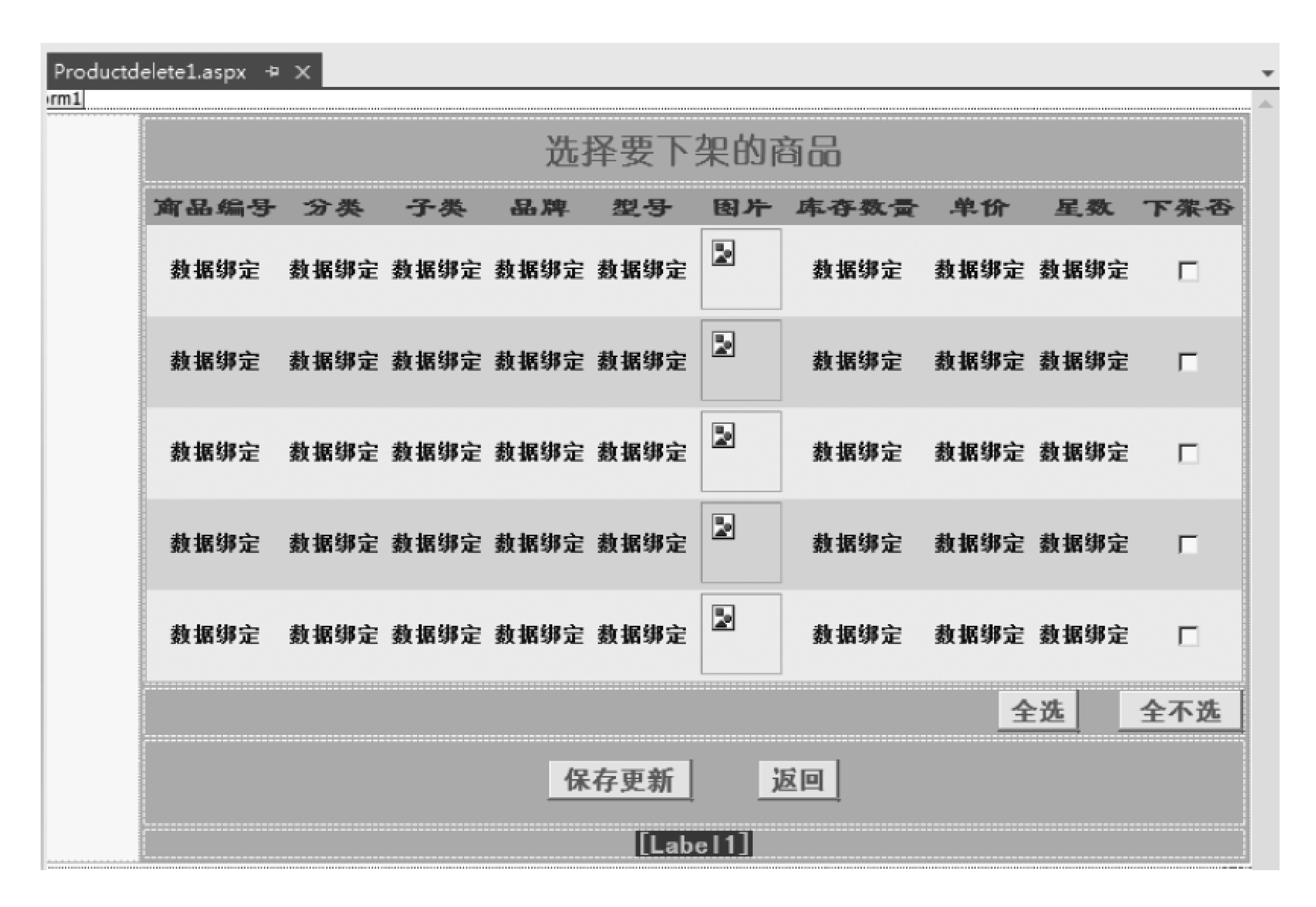


图 12.81 Productdelete1 网页的设计界面

```
<asp:TemplateField HeaderText = "下架否">
    <ItemTemplate>
        <asp:CheckBox ID = "CheckBox1" runat = "server" />
    </ItemTemplate>
    <HeaderStyle Font - Bold = "True" Font - Names = "隶书" Font - Size = "18px"
      ForeColor = "Blue" />
    <ItemStyle HorizontalAlign = "Center" />
</asp:TemplateField>
```

“全选”按钮 allsel 的单击事件处理方法如下：

```
protected void allsel_Click(object sender, EventArgs e)       //全部选中
{   int i;
    CheckBox chkbox;                                           //复选框对象
    for (i = 0; i < GridView1.Rows.Count; i++)
    {   chkbox = GridView1.Rows[i].FindControl("CheckBox1") as CheckBox;
        chkbox.Checked = true;
    }
}
```

“全不选”按钮 allnosel 的单击事件处理方法如下：

```
protected void allnosel_Click(object sender, EventArgs e)     //全部不选中
{   int i;
    CheckBox chkbox;                                           //复选框对象
    for (i = 0; i < GridView1.Rows.Count; i++)
    {   chkbox = GridView1.Rows[i].FindControl("CheckBox1") as CheckBox;
        chkbox.Checked = false;
    }
}
```

“保存更新”按钮 Button1 的单击事件处理方法如下：

```
protected void Button1_Click(object sender, EventArgs e)
{   string spno;                                               //商品编号
    CheckBox chkbox;                                           //复选框对象
    int i;
    for (i = 0; i < GridView1.Rows.Count; i++)
    {   spno = GridView1.Rows[i].Cells[0].Text;               //提取该行的商品编号
        chkbox = GridView1.Rows[i].FindControl("CheckBox1") as CheckBox;
        if (chkbox.Checked)
            Update(spno);                                      //调用自定义过程进行更新
    }
    Response.Redirect("~/dispinfo.aspx?info = 选择的下架商品信息已更新!");
}
protected void Update(string spno)  //自定义过程,用 UPDATE 语句修改商品信息
{   mysql = "UPDATE Products SET 有效否 = '0'"
      + " WHERE 商品编号 = '" + spno + "'";
    mydb.ExecuteNonQuery(mysql);
}
```

12.8.10 “按商品分类统计”功能网页设计

按商品分类统计由 Productst1 网页实现，其设计界面如图 12.82 所示，用于统计所有分类商品的销售数量和销售金额。其自定义的绑定方法如下：

```
public void bind()
{   mysql = "SELECT 分类,SUM(数量) as 销售数量,SUM(金额) as 销售金额 "
      + "FROM Sales "
      + "GROUP BY 分类";
    myds = mydb.ExecuteQuery(mysql, "Sales");
    GridView1.DataSource = myds.Tables["Sales"];
     GridView1.DataBind();      //在 GridView1 控件中显示满足查询条件的记录
}
```

Productst1.aspx

商品统计-分类

分类	销售数量	销售金额
数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定
数据绑定	数据绑定	数据绑定

1 2

返 回

图 12.82　Productst1 网页的设计界面

例如,管理员 system/manager 进入按商品分类统计功能,其结果如图 12.83 所示。

商品统计-分类

分类	销售数量	销售金额
电脑办公	143	646129
手机/数码	76	239980

返 回

图 12.83　Productst1 网页的设执行面

12.8.11　"按商品子类统计"功能网页设计

按商品子类统计由 Productst2 网页实现,其设计界面如图 12.84 所示,用于统计所有子类商品的销售数量和销售金额。其自定义的绑定方法如下:

```
public void bind()
{   mysql = "SELECT 分类,子类,SUM(数量) as 销售数量,SUM(金额) as 销售金额 "
      + "FROM Sales "
      + "GROUP BY 分类,子类";
    myds = mydb.ExecuteQuery(mysql, "Sales");
    GridView1.DataSource = myds.Tables["Sales"];
    GridView1.DataBind();                      //在 GridView1 控件中显示满足查询条件的记录
}
```

例如,管理员 system/manager 进入按商品分类统计功能,其结果如图 12.85 所示。



图 12.84　Productst2 网页的设计界面

商品统计—子类

分类	子类	销售数量	销售金额
电脑办公	笔记本	61	369421
手机/数码	单反数码相机	25	134189
手机/数码	手机	51	105791
电脑办公	台式机	82	276708

返 回

图 12.85　Productst2 网页的执行界面

12.8.12　“按商品品牌统计”功能网页设计

按商品品牌统计由 Productst3 网页实现，其设计界面如图 12.86 所示，用于统计所有品牌商品的销售数量和销售金额。其自定义的绑定方法如下：



图 12.86　Productst3 网页的设计界面

```
public void bind()
{   mysql = "SELECT 分类,子类,品牌,SUM(数量) as 销售数量,SUM(金额) as 销售金额 "
      + "FROM Sales "
      + "GROUP BY 分类,子类,品牌";
    myds = mydb.ExecuteQuery(mysql, "Sales");
    GridView1.DataSource = myds.Tables["Sales"];
    GridView1.DataBind();                        //在 GridView1 控件中显示满足查询条件的记录
}
```

例如,管理员 system/manager 进入按商品分类统计功能,其结果如图 12.87 所示。

商品统计-品牌

分类	子类	品牌	销售数量	销售金额
电脑办公	笔记本	ThinkPad	40	195430
电脑办公	笔记本	戴尔	21	173991
电脑办公	台式机	惠普	25	58980
电脑办公	台式机	联想	57	217728
手机/数码	单反数码相机	佳能	15	90589
手机/数码	单反数码相机	尼康	10	43600
手机/数码	手机	华为	40	95962
手机/数码	手机	小米	11	9829

返 回

图 12.87　Productst3 网页的执行界面

12.8.13　“设置顾客学历数据”功能网页设计

设置顾客学历数据功能由 setgkxl 和 setgkxl1 两个网页实现。

1. setgkxl 网页设计

setgkxl 网页用于显示 Education 表中所有学历数据,其设计界面如图 12.88 所示。



图 12.88　setgkxl 网页的设计界面

管理员可以在学历文本框中输入一个学历,单击“添加”按钮 Button2 将其插入到 Education 表中,学历编号是 SQL Server 自动生成的。对应的事件处理方法如下:

```
protected void Button2_Click(object sender, EventArgs e)
{   if (TextBox1.Text.Trim() != "")
    {   mysql = "SELECT * FROM Education WHERE 学历 = '" + TextBox1.Text.Trim() + "'";
        int i = mydb.Rownum(mysql);
        if (i > 0)
            Label1.Text = "提示：学历重复!";
        else
        {   mysql = "INSERT INTO Education(学历) VALUES('"
             + TextBox1.Text.Trim() + "')";
            mydb.ExecuteNonQuery(mysql);
            bind();
            Label1.Text = "";
        }
    }
    else
        Label1.Text = "必须输入一个学历!";
}
```

GridView1 控件中有两个 CommandField 字段，“编辑”字段的事件处理方法如下（调用 setgkxl1 网页）：

```
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{   //编辑记录
    e.Cancel = true;
    string xlbh;                                        //学历编号
    xlbh = GridView1.DataKeys[e.NewEditIndex].Value.ToString().Trim();
    string xl;                                          //学历
    xl = GridView1.Rows[e.NewEditIndex].Cells[1].Text.Trim();
    Response.Redirect("setgkxl1.aspx?xlbh=" + xlbh + "&xl=" + xl);      //转向 setgkxl1 网页
}
```

“删除”字段的事件处理方法如下：

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{   //删除记录
    e.Cancel = true;
    string xlbh;                                        //学历编号
    xlbh = GridView1.DataKeys[e.RowIndex].Value.ToString();
    mysql = "DELETE FROM Education WHERE 编号 = '" + xlbh + "'";
    mydb.ExecuteNonQuery(mysql);                        //执行 DELETE 语句
    bind();
}
```

2. setgkxl1 网页设计

setgkxl1 网页用于编辑单个学历数据，其设计界面如图 12.89 所示。其中，学历编号不能更改，只能更改学历。管理员单击“提交”按钮 Button1 后调用如下事件处理方法更新 Education 表：

```
protected void Button1_Click(object sender, EventArgs e)
{   mysql = "UPDATE Education SET 学历 = '" + TextBox2.Text.Trim()
         + "' WHERE 编号 = '"
         + TextBox1.Text.Trim() + "'";                  //构造 UPDATE 语句
    mydb.ExecuteNonQuery(mysql);                        //执行 UPDATE 语句
    Response.Redirect("setgkxl.aspx");                  //转向 setgkxl 网页
}
```



图 12.89　setgkxl1 网页的设计界面

12.8.14　“设置顾客地区数据”功能网页设计

设置顾客地区数据功能由 setgkdq 和 setgkdq1 两个网页实现。

1. setgkdq 网页设计

setgkdq 网页用于显示 Area 表中所有地区数据和添加地区数据，其设计界面如图 12.90 所示。设计思路与 setgkxl 网页相似。

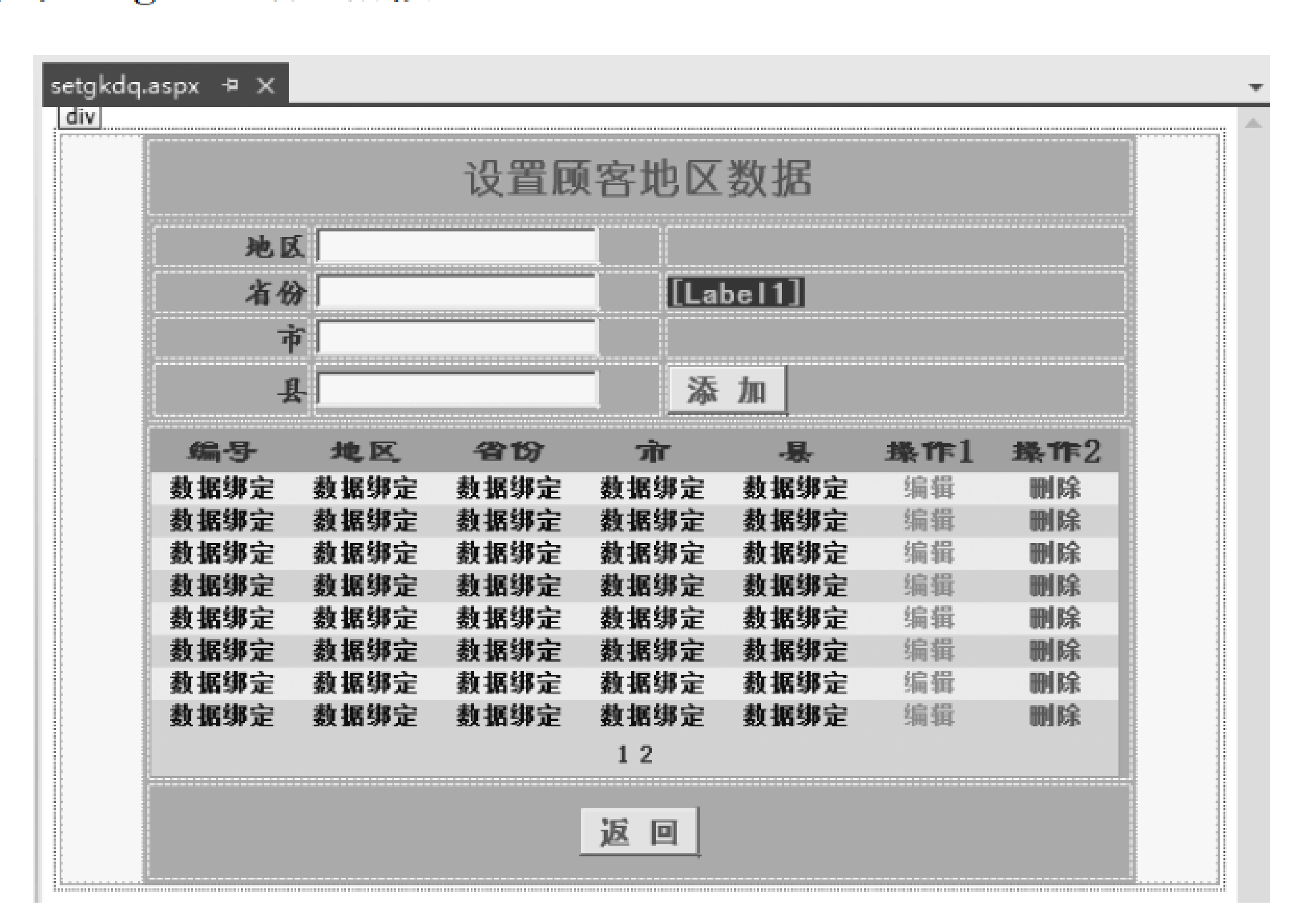


图 12.90　setgkdq 网页的设计界面

2. setgkdq1 网页设计

setgkdq1 网页用于编辑单个地区数据，其设计界面如图 12.91 所示。其中，地区编号不能更改，只能更改该编号的其他数据。设计思路与 setgkxl1 网页相似。管理员单击“提交”按钮 Button1 后将数据更新保存到 Area 表。



图 12.91　setgkdq1 网页的设计界面

12.8.15　“设置商品类型数据”功能网页设计

设置商品类型数据功能由 setsplx 和 setsplx1 两个网页实现。

1. setsplx 网页设计

setsplx 网页用于显示 ProdType 表中所有商品类型数据和添加类型数据，其设计界面如图 12.92 所示。设计思路与 setgkxl 网页相似。

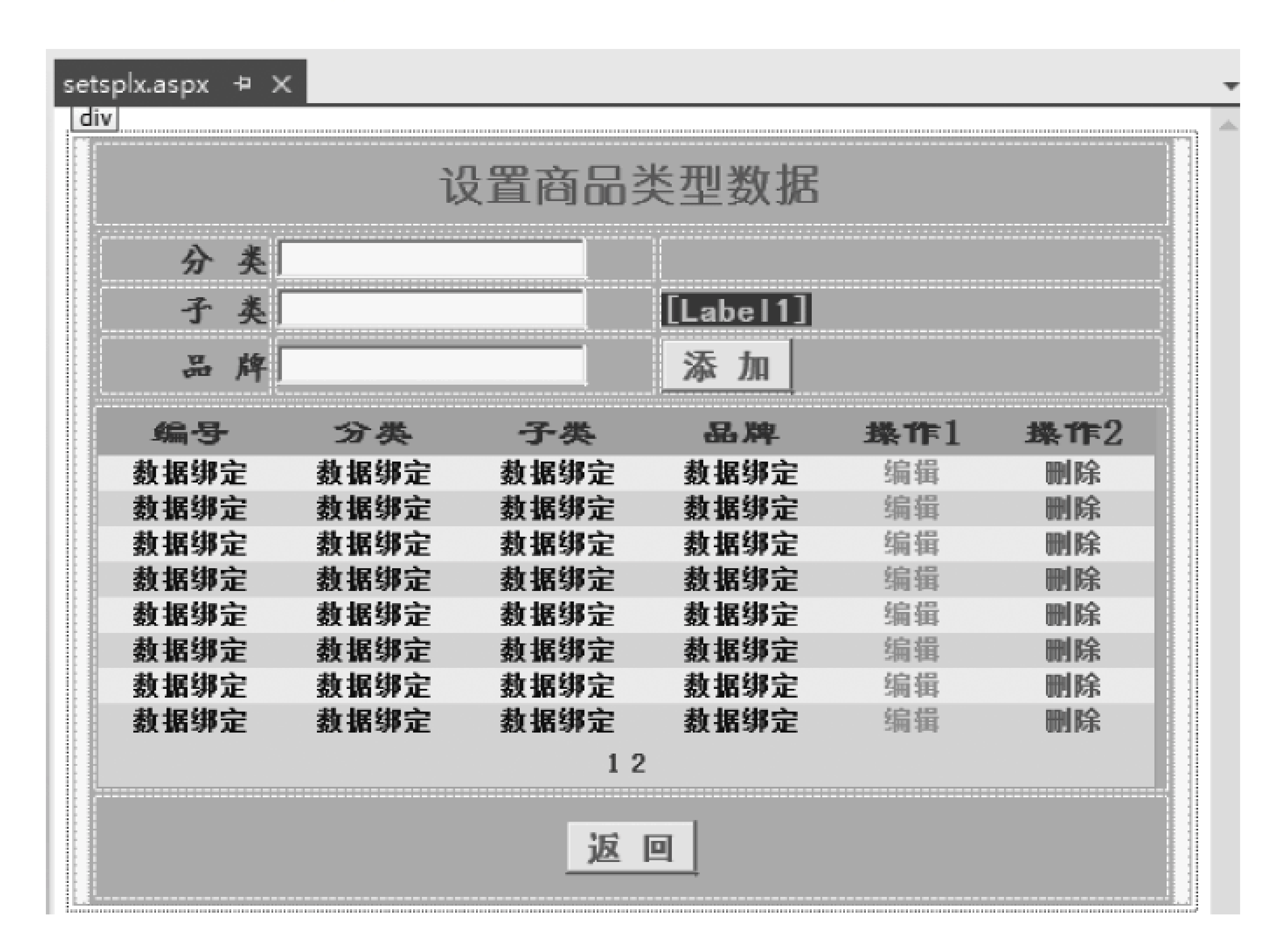


图 12.92　setsplx 网页的设计界面

2. setsplx1 网页设计

setsplx1 网页用于编辑单个商品类型数据，其设计界面如图 12.93 所示。其中，类型编号不能更改，只能更改该编号的其他数据。设计思路与 setgkxl1 网页相似。管理员单击“提交”按钮 Button1 后将数据更新保存到 ProdType 表。



图 12.93　setsplx1 网页的设计界面

12.8.16　“删除下架的商品信息”功能网页设计

前面的商品下架功能只是将 Products 表中该商品信息的“有效否”列设置为 0。删除下架功能就是将“有效否”为 0 的商品信息从 Products 表中删除。

实现删除下架的商品信息功能的网页为 deletexjproduct，其设计界面如图 12.94 所示。

主要包含一个 GridView1 控件。它首先从 Products 表中查找出所有“有效否”为 0 的商品信息，然后将结果在 GridView1 控件中显示。GridView1 控件中有一个“删除否”字段，其源视图代码如下：

```
<asp:TemplateField HeaderText = "删除否">
    <ItemTemplate>
        <asp:CheckBox ID = "CheckBox1" runat = "server" />
    </ItemTemplate>
    <HeaderStyle Font - Bold = "True" Font - Names = "隶书" Font - Size = "18px"
        ForeColor = "Blue" />
     <ItemStyle HorizontalAlign = "Center" />
</asp:TemplateField>
```



图 12.94　deletexjproduct 网页的设计界面

“确定删除”按钮 Button1 的单击事件处理方法如下：

```
protected void Button1_Click(object sender, EventArgs e)
{   string spno;                                              //商品编号
    CheckBox chkbox;                                          //复选框对象
    Image img;                                                //图像对象
    int i;
    for (i = 0; i < GridView1.Rows.Count; i++)
    {   spno = GridView1.Rows[i].Cells[0].Text;               //提取该行的商品编号
        chkbox = GridView1.Rows[i].FindControl("CheckBox1") as CheckBox;
        img = GridView1.Rows[i].FindControl("Image1") as Image;
        if (chkbox.Checked)
            Update(spno, img.ImageUrl);                       //调用自定义过程进行更新
    }
    Response.Redirect("~/dispinfo.aspx?info = 选择的下架商品信息已删除!");
}
protected void Update(string spno, string file)
//自定义过程,用 DELETE 语句删除商品信息及相应的图像文件
```

```
{   mysql = "DELETE Products " + " WHERE 商品编号 = '" + spno + "'";
    mydb.ExecuteNonQuery(mysql);
    string filename = Server.MapPath(file);
    if (File.Exists(filename))                    //若文件 filename 已存在
        File.Delete(filename);                    //删除 filename 文件
}
```

“全选”命令按钮 allsel 的单击事件处理方法如下：

```
protected void allsel_Click(object sender, EventArgs e)
{   int i;
    CheckBox chkbox;                              //复选框对象
    for (i = 0; i < GridView1.Rows.Count; i++)
    {   chkbox = GridView1.Rows[i].FindControl("CheckBox1") as CheckBox;
        chkbox.Checked = true;
    }
}
```

“全不选”命令按钮 allnosel 的单击事件处理方法如下：

```
protected void allnosel_Click(object sender, EventArgs e)
{   int i;
    CheckBox chkbox;                              //复选框对象
    for (i = 0; i < GridView1.Rows.Count; i++)
    {   chkbox = GridView1.Rows[i].FindControl("CheckBox1") as CheckBox;
        chkbox.Checked = false;
    }
}
```

12.8.17　“系统初始化”功能网页设计

系统初始化功能对应的网页为 systeminit，其设计界面如图 12.95 所示。当管理员单击“确定”按钮 Button1 时实现系统初始化，删除所有数据，仅仅在 Users 表中保留一个管理员级别的用户 system/manager。

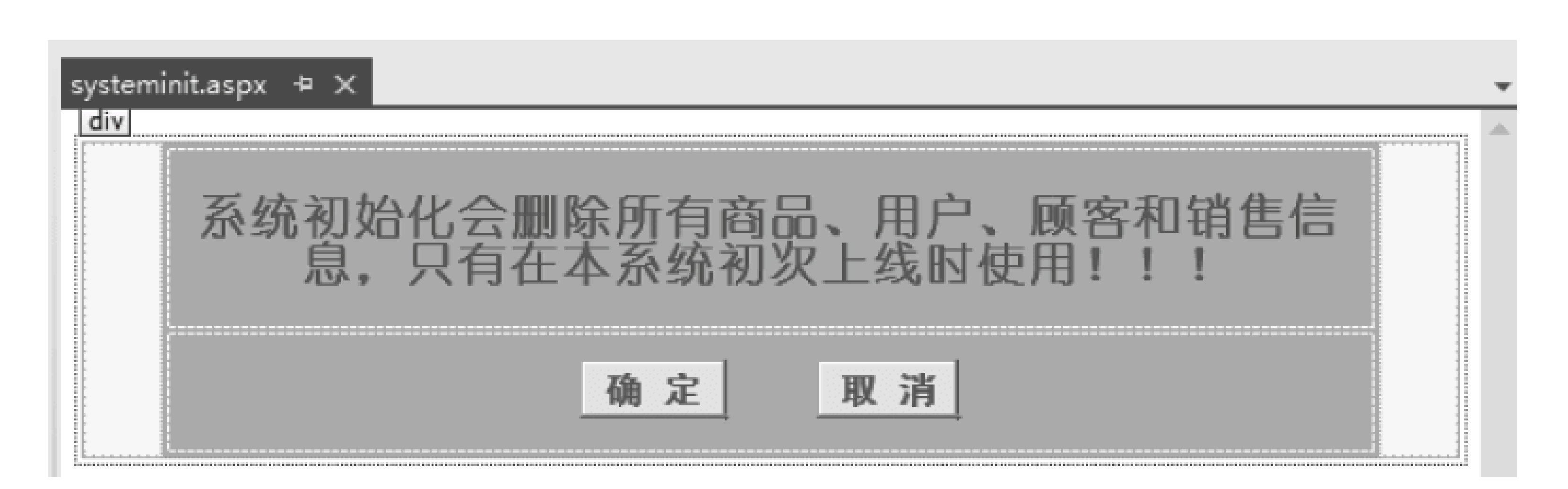


图 12.95　systeminit 网页的设计界面

系统初始化的事件处理方法如下：

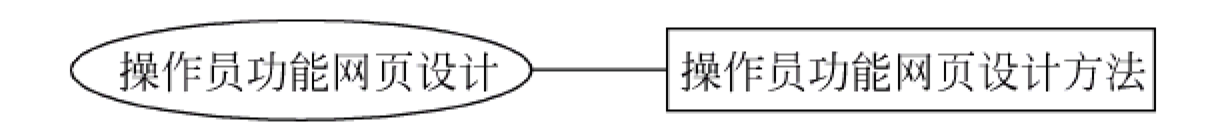
```
protected void Button1_Click(object sender, EventArgs e)
{   deltable("Area");
    deltable("Comment");
    deltable("Customers");
    deltable("Education");
    deltable("OrderForm");
```

```
        deltable("ProdType");
        deltable("Products");
        deltable("Sales");
        deltable("ShoppingCart");
        delusertable("Users");
        delpicture();                                    //删除 Picture 文件夹中所有图像文件
        Response.Redirect("~/dispinfo.aspx?info=系统初始化完毕!"
          + "下次以 system/manager 管理员身份登录");
    }
    protected void deltable(string tname)                  //删除 tname 表中所有记录
    {   CommDB mydb = new CommDB();                        //创建 CommDB 类对象
        string mysql;
        mysql = "DELETE " + tname;
        mydb.ExecuteNonQuery(mysql);
    }
    protected void delusertable(string tname)              //删除用户表中记录
    {   CommDB mydb = new CommDB();                        //创建 CommDB 类对象
        string mysql;
        mysql = "DELETE " + tname;
        mydb.ExecuteNonQuery(mysql);
        mysql = "INSERT INTO Users(用户名,密码,类型,有效否) "
          + "VALUES('system','manager','管理员',1)";
        mydb.ExecuteNonQuery(mysql);
    }
    protected void delpicture()                           //删除 Picture 文件夹中所有图像文件
    {   string path = Server.MapPath("~/Picture");
        if (Directory.GetFileSystemEntries(path).Length > 0)
        {   //遍历文件夹中所有文件
            foreach (string file in Directory.GetFiles(path))
            {   if (File.Exists(file))                     //若文件 file 已存在
                    File.Delete(file);                     //删除 file 文件
            }
        }
    }
```

另外,管理员更改我的密码功能对应的网页为 updatemanagerpass,它与 updatecustomerpass 网页几乎相同,这里不再介绍。

12.9　操作员功能网页设计

知识梳理



12.9.1　操作员功能主页设计

当操作员进入网站后,首先显示操作员功能主页 operatormenu,其设计界面如图 12.96 所示。它是基于母版页 MasterPage.master 的。该网页位于网站根目录下,它所调用的所有网页都放置在子 Operator 目录中。



图 12.96　operatormenu 网页的设计界面

operatormenu 网页的设计思路与游客功能主页 touristmenu 的设计思路相似，只是在 TreeView1 控件中列出操作员的功能项。

12.9.2　“添加新型号商品信息”功能主页设计

添加新型号商品信息功能对应的网页为 addnewProduct，其设计界面如图 12.97 所示。其中，FileUpload1 控件用于上传商品的图像文件。



图 12.97　addnewProduct 网页的设计界面

操作员输入正确的数据后，单击“确定”按钮 Button1，引发的事件处理方法如下：

```
protected void Button1_Click(object sender, EventArgs e)
{   if (Page.IsValid)
    {   int i;
```

```
            mysql = "SELECT * FROM Products WHERE 商品编号 = '" + bhTextBox.Text + "'";
            i = mydb.Rownum(mysql);
            if (i > 0)
                Label1.Text = "商品编号重复,不能添加该商品记录!";
            else
            {   string filestr;                                  //商品图片文件名
                if (FileUpload1.HasFile)
                {   filestr = Server.MapPath("/") + "\\Picture\\"
                     + FileUpload1.PostedFile.FileName;
                    try
                    {   FileUpload1.PostedFile.SaveAs(filestr);
                        Label1.Text = "提示: 文件成功上传到"
                         + FileUpload1.PostedFile.FileName;
                    }
                    catch (Exception ex)
                    {
                        Label1.Text = "提示: 文件上传失败," + ex.Message;
                    }
                }
                else
                {   Label1.Text = "提示: 没有指定任何要上传的图片文件";
                    return;
                }
                mysql = "INSERT INTO Products(商品编号,分类,子类,品牌,型号,"
                   + "单价,库存数量,图片,有效否,星数,评论数) VALUES('"
                   + bhTextBox.Text.Trim() + "','"
                   + DropDownList1.SelectedValue.ToString().Trim() + "','"
                   + DropDownList2.SelectedValue.ToString().Trim() + "','"
                   + DropDownList3.SelectedValue.ToString().Trim() + "','"
                   + xhTextBox.Text.Trim() + "',"
                   + priceTextBox.Text.Trim() + ","
                   + numTextBox.Text.Trim() + ",'"
                   + "~//Picture//" + FileUpload1.PostedFile.FileName.Trim() + "','"
                   + "1',0,0)";
                mydb.ExecuteNonQuery(mysql);
                Response.Redirect("~/dispinfo.aspx?info = 新型号商品已成功添加!");
            }
        }
        else
            Label1.Text = "提示: 商品信息错误,不能添加";
    }
```

例如,一个操作员添加 1111 编号的商品的界面如图 12.98 所示。

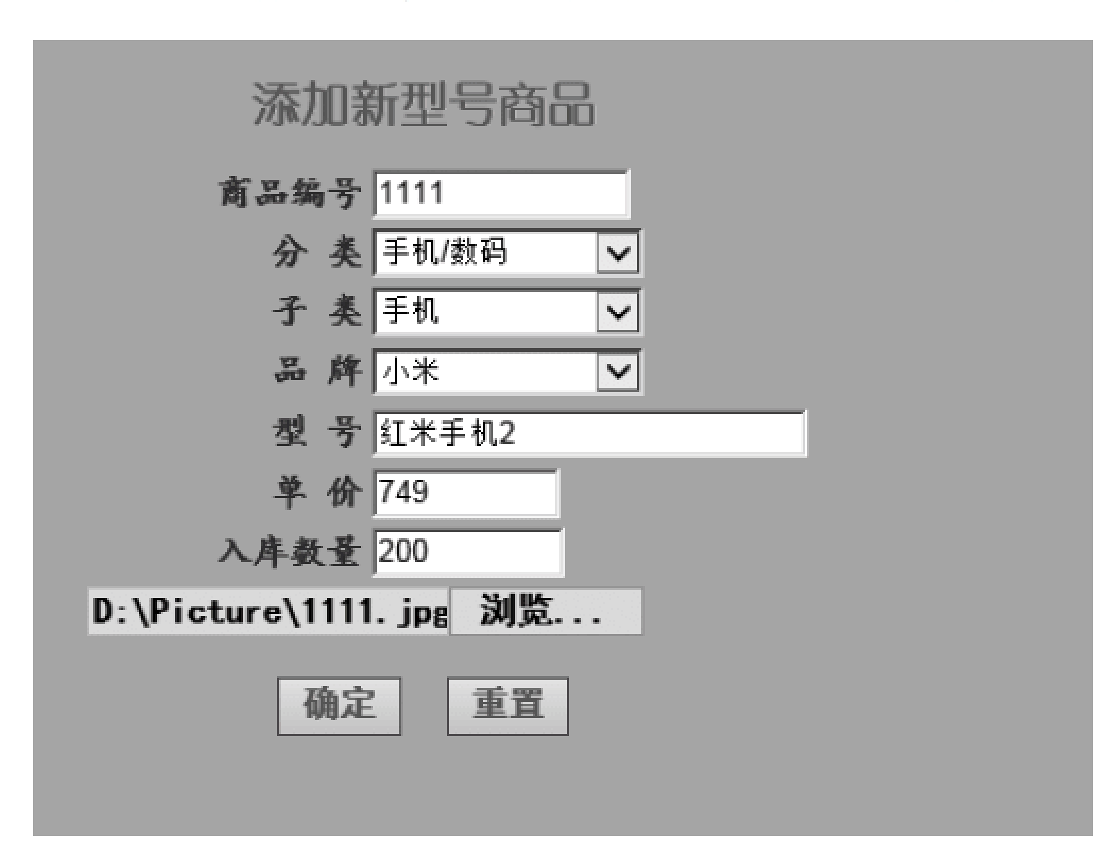


图 12.98 addnewProduct 网页的执行界面

说明：在建立商品库时，先将商品编号等信息和对应的图像文件准备好。为了方便，最好将商品图像文件名称与商品编号一致，并且将所有商品图像文件放在本地的一个目录中，这里放在 D:\Picture 目录中。商品图像文件的格式应是浏览器可以显示的图像格式，如.jpg 格式，否则还需要安装显示图像文件的插件。

12.9.3　"更新老商品信息"功能主页设计

更新老商品信息功能由 updateoldProduct 和 updateoldProduct1 两个网页实现。

1. updateoldProduct 网页设计

updateoldProduct 网页和 dispcustomer 网页相似，其设计界面如图 12.99 所示，它用于设置查找更新老商品的条件，构成一个条件字符串 condstr，继而生成一个 SELECT 语句，存放在 Session["sql"]会话中，再转向 updateoldProduct1 网页。



图 12.99　updateoldProduct 网页的设计界面

2. updateoldProduct1 网页设计

updateoldProduct1 网页的设计界面如图 12.100 所示，主要包含一个 GridView1 控件。它首先执行 Session["sql"]中的 SELECT 语句，然后将结果在 GridView1 控件中显示。GridView1 控件中添加有"单价"和"增加库存"两个字段，其源视图代码如下：

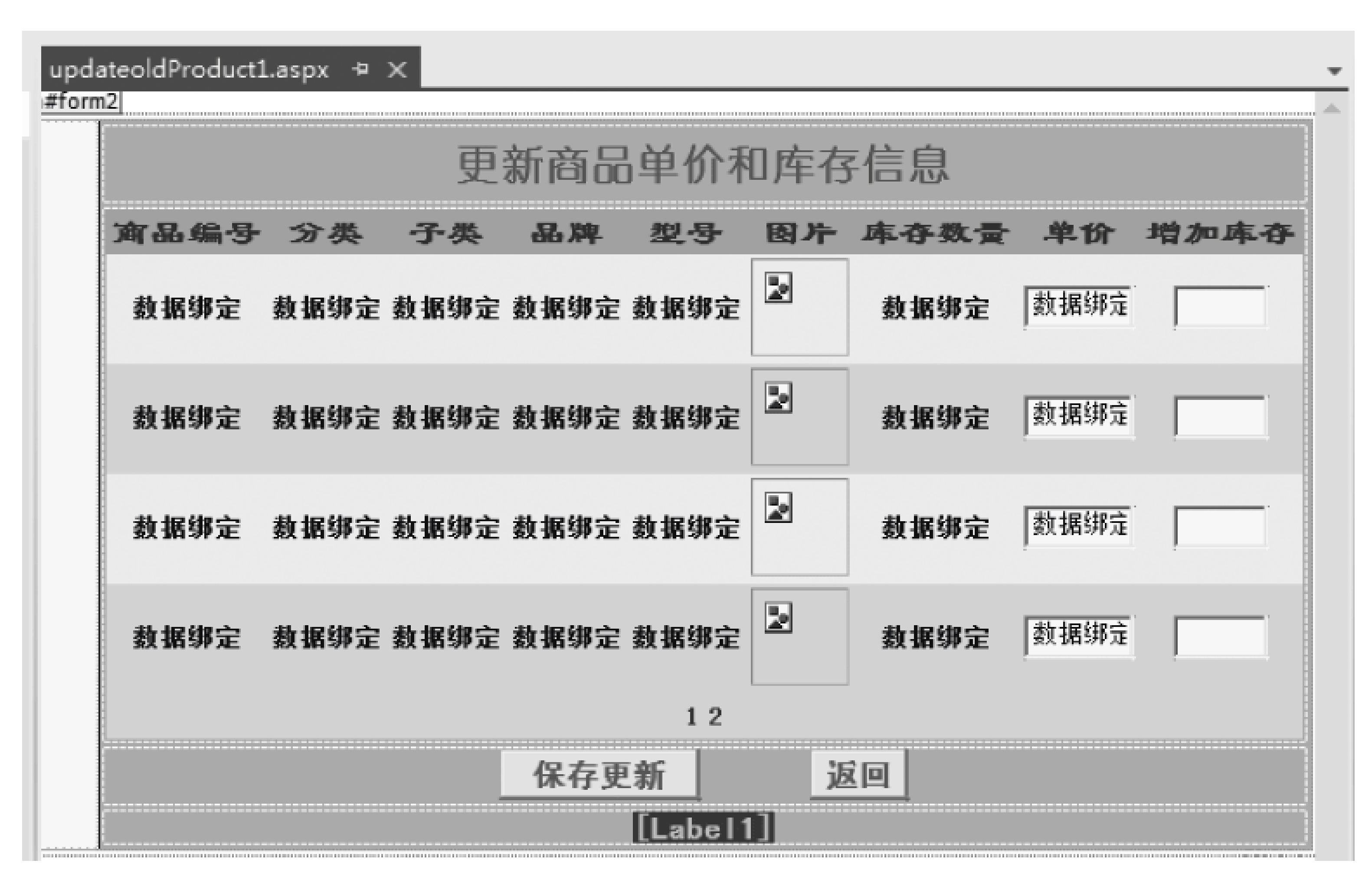


图 12.100　updateoldProduct1 网页的设计界面

```
<asp:TemplateField HeaderText = "单价">
     <ItemTemplate>
        <asp:TextBox ID = "TextBox1" runat = "server" Height = "16px"
          Text = '<% # DataBinder.Eval(Container.DataItem,"单价") %>'
          Width = "48px"></asp:TextBox>
    </ItemTemplate>
    <HeaderStyle Font - Bold = "True" Font - Names = "隶书" Font - Size = "18px"
        ForeColor = "Blue" />
    <ItemStyle HorizontalAlign = "Center" />
</asp:TemplateField>
<asp:TemplateField HeaderText = "增加库存">
    <ItemTemplate>
       <asp:TextBox ID = "TextBox2" runat = "server" Height = "16px"
          Width = "40px"></asp:TextBox>
    </ItemTemplate>
    <HeaderStyle Font - Bold = "True" Font - Size = "18px" Font - Names = "隶书"
       ForeColor = "Blue" />
    <ItemStyle HorizontalAlign = "Center" />
</asp:TemplateField>
```

操作员可以输入“单价”和“增加库存”两个字段值，其他字段不能更新。

当操作员单价“保存更新”按钮 Button1 时，执行如下事件处理方法实现商品信息的更新：

```
protected void Button1_Click(object sender, EventArgs e)
{                                                       //保存更新
    savedata();
    Label1.Text  =  "当前页的商品更新已保存";
}
protected void savedata()                               //自定义过程,保存商品更新
{   string spno;                                        //商品编号
    TextBox djtxt;                                      //单价文本框
    TextBox addkctxt;                                   //增加库存文本框
    int i;
    for (i  =  0; i < GridView1.Rows.Count; i++)
    {   spno  =  GridView1.Rows[i].Cells[0].Text;       //提取该行的商品编号
        djtxt  =  GridView1.Rows[i].FindControl("TextBox1") as TextBox;
                //在该行中找 TextBox1 控件
        addkctxt  =  GridView1.Rows[i].FindControl("TextBox2") as TextBox;
                //在该行中找 TextBox2 控件
        Update(spno, djtxt.Text.Trim(),addkctxt.Text.Trim());
                //调用自定义过程进行更新
    }
}
protected void Update(string spno, string dj,string addkc)
//自定义过程,用 UPDATE 语句修改商品信息
{   mysql  =  "UPDATE Products SET 单价 = "  +  dj  +  ",库存数量 = 库存数量 + "
     +  addkc  +  " WHERE 商品编号  =  '"  +  spno  +  "'";
    mydb.ExecuteNonQuery(mysql);
}
```

12.9.4　“查看新订单”功能主页设计

查看新订单功能对应的网页为 dispnewod，其设计界面如图 12.101 所示。它在 GridView1

控件中显示 OrderForm 表中使用处理否列为 0 的订单记录。



图 12.101　dispnewod 网页的设计界面

当操作员单击“转换为 Excel 文件”按钮 Button1 时，执行如下事件处理方法产生包含所有未处理订单的 Excel 文件：

```
protected void Button1_Click(object sender, EventArgs e)
{
    export("abc.xls");
}
protected void export(string filename)
{   string mysql1;
    DataSet myds1 = new DataSet();                          //创建 DataSet 对象
    HttpResponse resp;
    resp = Page.Response;
    resp.ContentEncoding = System.Text.Encoding.GetEncoding("GB2312");
    resp.AppendHeader("Content-Disposition", "attachment;filename=" + filename);
    string colHeaders = "", ls_item = "";
    string colHeaders1 = "", ls_item1 = "";
    //定义表对象与行对象,同时用 DataSet 对其值进行初始化
    mysql = "SELECT * "
     + " FROM OrderForm"
     + " WHERE 处理否=0 ORDER BY 订单号";
    myds = mydb.ExecuteQuery(mysql, "OrderForm");
    DataTable dt = myds.Tables["OrderForm"];
    DataRow[] myRow = dt.Select();
    int i = 0, j=0,cl1;
    int cl = dt.Columns.Count;                              //主表(订单表)列数
    //取得主数据表各列标题,各标题之间以\t 分割,最后一个列标题后加回车符\n
    for (i = 0; i<cl; i++)
    {   if (i == (cl - 1))                                  //最后一列,加\n
            colHeaders += dt.Columns[i].Caption.ToString() + "\n";
        else
            colHeaders += dt.Columns[i].Caption.ToString() + "\t";
    }
    foreach (DataRow row in myRow)                          //逐行处理主表的数据
    {   resp.Write(colHeaders);                             //输出主表的标题信息
        //当前行数据写入 HTTP 输出流,并且置空 ls_item 以便下行数据
        ls_item = "";
```

```
            for (i = 0; i < cl; i++)
            {   if (i == (cl - 1))                          //最后一列,加\n
                    ls_item += row[i].ToString() + "\n";
                else
                    ls_item += row[i].ToString() + "\t";
            }
            resp.Write(ls_item);                            //输出主表的一行
            mysql1 = "SELECT * FROM Sales WHERE 订单号 = " + row[0].ToString();
            Label1.Text += row[0].ToString() + " ";
            myds1 = mydb.ExecuteQuery(mysql1, "Sales");
            DataTable dt1 = myds1.Tables["Sales"];
            DataRow[] myRow1 = dt1.Select();
            cl1 = dt1.Columns.Count;                        //子表列数
            //取得数据表各列标题,各标题之间以\t 分割,最后一个列标题后加回车符\n
            colHeaders1 = "";
            for (j = 0; j < cl1; j++)
            {   if (j == (cl1 - 1))                         //最后一列,加\n
                    colHeaders1 += dt1.Columns[j].Caption.ToString() + "\n";
                else
                    colHeaders1 += dt1.Columns[j].Caption.ToString() + "\t";
            }
            resp.Write(colHeaders1);                        //输出子表的标题信息
            foreach (DataRow row1 in myRow1)                //逐行处理子表数据
            {   ls_item1 = "";
                //当前行数据写入 HTTP 输出流,并且置空 ls_item1 以便下行数据
                for (j = 0; j < cl1; j++)
                {   if (j == (cl1 - 1))                     //最后一列,加\n
                        ls_item1 += row1[j].ToString() + "\n";
                    else
                        ls_item1 += row1[j].ToString() + "\t";
                }
                resp.Write(ls_item1);
            }
        }
        resp.End();
    }
```

例如,操作员进入查看新订单功能,如图 12.102 所示,单击“转换为 Excel 文件”按钮,网页下方出现“要打开或保存来自 localhost 的 abc.xls 吗?”对话框,单击“打开”按钮,生成的 abc.xls 文件如图 12.103 所示,可以另存储到其他文件中或打印,然后交给快递员发货。

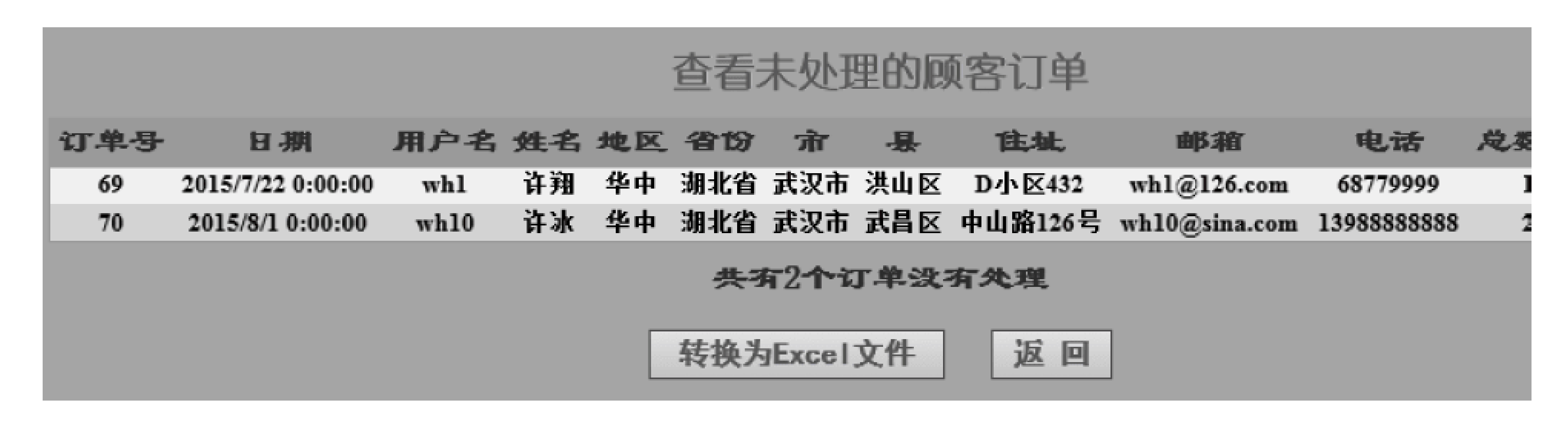


图 12.102 dispnewod 网页的执行界面



图 12.103　生成的 abc.xls 文件

12.9.5 "新订单处理"功能主页设计

新订单处理功能由 newodpross 和 newodpross1 两个网页实现。该功能是操作员向订单顾客发送邮件，表明订单已经处理。

1. newodpross 网页设计

newodpross 网页显示所有尚未处理的新订单，其设计界面如图 12.104 所示。当操作员单击某个订单的"选择"按钮时，执行如下事件处理方法产生邮件内容并指向 newodpross1 网页：



图 12.104　newodpross 网页的设计界面

```
protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
{   string yjlr = "";
    string yx  =  GridView1.SelectedRow.Cells[9].Text.ToString().Trim();
            //提取所选行的邮箱
    Session["yx"] = yx;
    string ddh  =  GridView1.SelectedRow.Cells[0].Text.ToString().Trim();
            //提取所选行的订单号
    Session["ddh"] = ddh;
    string rq  =  GridView1.SelectedRow.Cells[1].Text.ToString().Trim();
            //提取所选行的日期
    string name  =  GridView1.SelectedRow.Cells[3].Text.ToString().Trim();
```

```
            //提取所选行的姓名
    string sf = GridView1.SelectedRow.Cells[5].Text.ToString().Trim();
            //提取所选行的省份
    string city = GridView1.SelectedRow.Cells[6].Text.ToString().Trim();
            //提取所选行的城市
    string xm = GridView1.SelectedRow.Cells[7].Text.ToString().Trim();
            //提取所选行的县
    string zz = GridView1.SelectedRow.Cells[8].Text.ToString().Trim();
            //提取所选行的住址
    string dh = GridView1.SelectedRow.Cells[10].Text.ToString().Trim();
            //提取所选行的电话
    string zsl = GridView1.SelectedRow.Cells[11].Text.ToString().Trim();
            //提取所选行的总数量
    string zjr = GridView1.SelectedRow.Cells[12].Text.ToString().Trim();
            //提取所选行的总金额
    yjlr += "亲爱的" + name + ":\r\n";
    yjlr += "   你于" + rq + "在本商务网站下了订单\r\n";
    yjlr += "  共订购了" + zsl + "件商品,总金额是" + zjr + "元\r\n";
    yjlr += "  订单号是:" + ddh + "\r\n";
    yjlr += "  收件人地址是: " + sf + city + xm + zz + "\r\n";
    yjlr += "  收件人电话是:" + dh + "\r\n";
    yjlr += "请你注意查收快件并货到付款,谢谢!!!,欢迎下次光临";
    Session["yjlr"] = yjlr;
    Response.Redirect("newodpross1.aspx");
}
```

2. newodpross1 网页设计

newodpross1 网页显示邮件内容，其设计界面如图 12.105 所示。当操作员单击“发送”按钮时，执行如下事件处理方法发送该邮件：



图 12.105 newodpross1 网页的设计界面

```
protected void Button1_Click(object sender, EventArgs e)
{   MailMessage mail = new MailMessage();
    mail.To.Add(new MailAddress(TextBox1.Text.Trim()));
    mail.From = new MailAddress("abc@126.com");
    mail.Subject = TextBox2.Text;
    mail.Body = TextBox3.Text.Trim();
```

```
        mail.Priority = MailPriority.High;
        SmtpClient smtpmail = new SmtpClient();
        smtpmail.DeliveryMethod = SmtpDeliveryMethod.Network;
                                                    //指定电子邮件发送方式为 Network
        smtpmail.Host = "smtp.126.com";             //指定 smtp 服务器地址
        smtpmail.Credentials =
            new System.Net.NetworkCredential("abc", "1234");    //认证
        try
        {
            smtpmail.Send(mail);
        }
        catch (Exception ex)
        {
            Label1.Text = "出错提示: " + ex.Message + ",请按返回按钮";
        }
        string mysql;                               //SQL 表达式
        CommDB mydb = new CommDB();                 //创建 CommDB 类对象
        mysql = "UPDATE OrderForm SET 处理否 = 1 WHERE 订单号 = '"
         + Session["ddh"].ToString().Trim() + "'";
        mydb.ExecuteNonQuery(mysql);
        Response.Redirect("newodpross.aspx");
    }
```

其中,abc@126.com 是本网站服务商注册的服务邮箱,密码为 1234,它需要邮箱服务商认证。

例如,操作员发给 wh10 顾客的邮件如图 12.106 所示。单击"发送"按钮即可发送该邮件(邮件中的内容是由本网站自动生成的)。

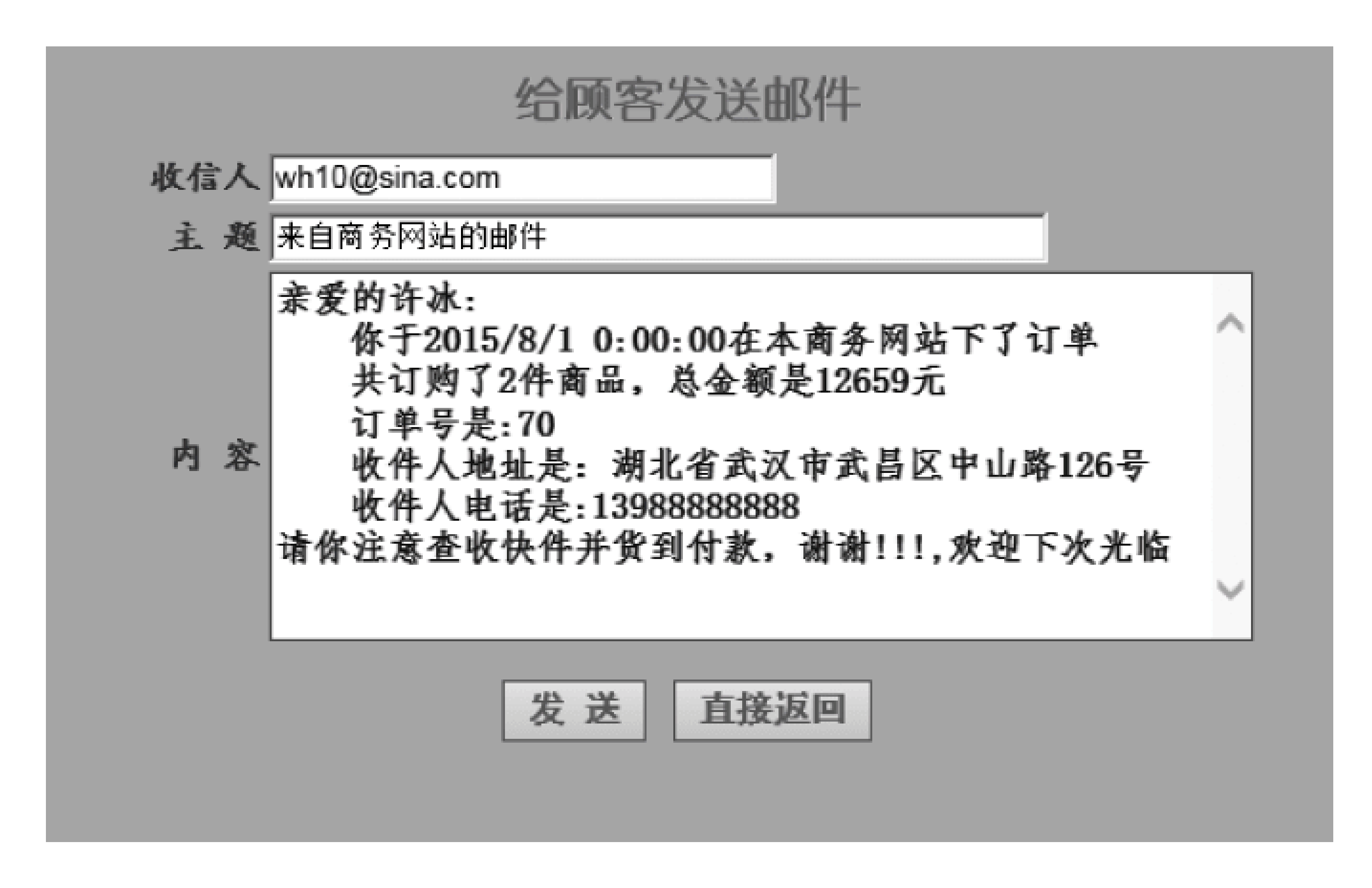


图 12.106　newodpross1 网页的执行界面

12.9.6　"新订单结算处理"功能主页设计

当快递员已经将货物送给顾客,并收款后通知操作员,操作员进行收货处理,即新订单结算处理,对应的网页为 receiptprocess。其设计界面如图 12.107 所示。

界面中通过一个 GridView1 控件显示 OrderForm 表中所有"结算否"列为 0 的订单记录,操作员勾选所有需要结算的订单后的复选框。单击"确定"按钮。

本网页的设计思路与 12.8.16 节的 deletexjproduct 网页设计相似。



图 12.107　receiptprocess 网页的设计界面

另外，操作员更改我的密码功能对应的网页为 updateoperatorpass，它与 updatecustomerpass 网页几乎相同，这里不再介绍。

附录A　部分练习题参考答案

第1章练习题

1. 单项选择题

(1) C	(2) B	(3) D	(4) B	(5) B
(6) A	(7) B	(8) A	(9) A	(10) B
(11) D	(12) C	(13) B	(14) A	

2. 问答题

(1)【答】 电子商务是指交易当事人或参与人利用计算机技术和网络技术等现代信息技术所进行的各类商务活动。和传统商务比较,电子商务的交易对象是面向全球的,交易时间是24小时,以及它的营销活动是一对一,而且它的销售地点是一个虚拟空间,顾客可以按自己的方式自由购物,另外,企业能迅速捕捉顾客的需求,及时应对;此外,在具体运作过程中,电子商务也是不同的,它交易前的准备是通过交易双方的网址和主页完成的,在贸易磋商过程中是用电子化的记录,文件或报文在网上传递,而交易双方签订的是电子合同,其支付方式是用信用卡、电子支票、电子现金和电子钱包。

(4)【答】 静态网页以HTML源文件的形式存储在服务器端的存储设备上,当服务器接收到浏览器的页面请求时,服务器直接从存储设备上找到相应的HTML源文件,发给浏览器。动态网页在服务器端不直接存储,当服务器接收到浏览器的页面请求时,服务器启动特定的程序代码,动态生成相应的HTML网页文件,然后发送给浏览器。

第2章练习题

1. 单项选择题

(1) A	(2) C	(3) B	(4) C	(5) B
(6) D	(7) B	(8) D	(9) B	(10) C

第3章练习题

1. 单项选择题

(1) D	(2) D	(3) A	(4) B	(5) A
(6) B	(7) C	(8) C	(9) A	(10) C
(11) A	(12) B	(13) C	(14) B	(15) D
(16) A				

2. 问答题

(4)【答】 `<input name="TextBox1" type="text" id="TextBox1" />`

第 4 章练习题

1. 单项选择题

(1) D	(2) A	(3) D	(4) D	(5) A
(6) C	(7) C	(8) D	(9) C	(10) D
(11) C	(12) C	(13) D	(14) B	(15) D
(16) D	(17) D	(18) C	(19) A	(20) C

第 5 章练习题

1. 单项选择题

(1) D	(2) C	(3) B	(4) B	(5) B
(6) D	(7) B	(8) D	(9) C	(10) D
(11) D	(12) A	(13) C	(14) D	(15) D
(16) C	(17) B	(18) C	(19) B	(20) D
(21) A	(22) C			

2. 问答题

(3)【答】 弹出警告框,显示 403070。

(4)【答】 fun 函数如下:

```
function fun(obj)
{   var mystr = "显示文本:" + obj.options[obj.selectedIndex].text
        + ",值:" + obj.options[obj.selectedIndex].value;
    alert(mystr);
}
```

(5)【答】 foo 函数如下:

```
function foo()
{   var rg = document.getElementsByName("radioGroup");
    for (var i = 0; i < rg.length; i++)
    {   if (rg[i].checked)
            alert("你选择了第" + (i + 1) + "个选项");
    }
}
```

(6)【答】 JavaScript 函数如下:

```
function find(x)
{
    return x % 3 == 0 && x % 5 == 0 && x % 7 == 0;
}
function disp()
{   var mystr = "";
    var n, nb = 0;
    for (n = 1; n < 1000; n++)
    {   if (find(n))
        {   if (nb % 6 > 0) mystr += ",";
            nb++;
```

```
            mystr + = n;
            if (nb % 6 == 0) mystr += "\n";
        }
    }
    mystr + = "\n 共有" + nb + "个数";
    document.getElementById("textarea1").value = mystr;
}
```

(7)【答】 JavaScript 函数 setnull 如下：

```
function setnull() {
    var txts = document.getElementsByTagName("input");
    for (var i = 0;i < txts.length;i++)
        if (txts[i].type == "text")
            txts[i].value = "";
}
```

第 6 章练习题

1. 单项选择题

(1) B	(2) C	(3) C	(4) D	(5) A
(6) A	(7) B	(8) C	(9) C	(10) B
(11) C	(12) B	(13) C	(14) D	(15) B

2. 问答题

(1)【答】 从值类型接口转换到引用类型装箱。从引用类型转换到值类型拆箱。

(2)【答】 string str = null 是不给 str 变量分配内存空间,而 string str = ""是给 str 分配长度为空字符串的内存空间。

(4)【答】 private 表示私有成员,在类的内部才可以访问。protected 表示保护成员,该类内部和继承类中可以访问。public 表示公共成员,完全公开,没有访问限制。internal 表示在同一命名空间内可以访问。

(5)【答】 会执行,在 return 前执行。

(6)【答】 重载是方法的名称相同,参数或参数类型不同,进行多次重载以适应不同的需要。override 是进行基类中函数的重写,为了适应多态性。

(7)【答】 委托可以把一个方法作为参数代入另一个方法。委托可以理解为指向一个函数的引用。事件是一种特殊的委托。

(8)【答】 Button1_Click 事件处理方法如下：

```
protected void Button1_Click(object sender, EventArgs e)
{   int n = 2,i;
    string mystr = "";
    bool isprime;
    while (n <= 100)
    {   isprime = true;
        i = 2;
        while (i < n)
        {   if (n % i == 0)
            {   isprime = false;
                break;
```

```
            }
            i++;
        }
        if (isprime)
            mystr += n + " ";
        n++;
    }
    Label1.Text = mystr;
}
```

(9)【答】 Button1_Click 事件处理方法如下：

```
protected void Button1_Click(object sender, EventArgs e)
{   ArrayList myList = new ArrayList();
    Random rnd = new Random();
    while (myList.Count < 10)
    {   int num = rnd.Next(1, 101);
        if (!myList.Contains(num))
            myList.Add(num);
    }
    Label1.Text = "";
    foreach (int item in myList)
    Label1.Text += item.ToString() + " ";
}
```

(10)【答】 ①类 A 中的成员 a 是属性成员。
② 显示结果为 a＝1,b＝2。

第 7 章练习题

1. 单项选择题

(1) D	(2) A	(3) A	(4) C	(5) D
(6) C	(7) D	(8) A	(9) A	(10) D
(11) D	(12) D	(13) C	(14) C	(15) B
(16) A	(17) C	(18) B	(19) D	(20) A
(21) A	(22) A	(23) B	(24) C	(25) D
(26) B	(27) A	(28) D	(29) D	(30) B

2. 问答题

(1)【答】 服务器控件是指在服务器上执行程序逻辑的组件,通常具有一定的用户界面。服务器控件包含在 ASP .NET 网页中,当执行网页时,用户与控件发生交互行为；当网页提交时,控件可在服务器端引发事件,根据相关事件处理程序来进行事件处理。

(2)【答】 服务器控件的特点是具有 runat＝"server"属性。当 ASP .NET 网页被执行时,会检查网页上的标记有无 runat＝"server"属性,如果没有就会被直接发送到客户端的浏览器进行解析,如果有则表示这个控件可以被 ASP .NET 引擎所控制,需要等到程序执行完毕再将 HTML 控件的执行结果发送到客户端浏览器。

(3)【答】 HTML 服务器控件在服务器端执行完成后,发送到客户端的是生成的 HTML 元素。

(5)【答】 将其 AutoPostBack 属性设置为 true 即可。

(6)【答】 ASP .NET 中有这些验证控件：RequiredFieldValidator(非空验证)控件用于检查是否有输入值；CompareValidator(比较验证)控件用于按设定比较两个输入是否相同；RangeValidator(范围验证)控件用于验证输入是否在指定范围；RegularExpressionValidator(正则表达式验证)控件用于验证是否与指定的正则表达式匹配；CustomValidator(自定义验证)控件用于验证是否满足指定的条件；ValidationSummary(验证总结)控件用于集中验证信息处理。

第 8 章练习题

1. 单项选择题

(1) D	(2) D	(3) A	(4) A	(5) D
(6) D	(7) A	(8) C	(9) A	(10) C
(11) B	(12) A	(13) C	(14) D	(15) D
(16) B	(17) A	(18) B	(19) B	(20) A
(21) D	(22) D			

2. 问答题

(1)【答】 ASP .NET 中的内置对象有 Page 对象、Response 对象、Request 对象、Server 对象、Application 对象、Session 对象、Cookie 对象等。

(2)【答】 Page. IsPostBack 属性和 Page. IsValid 属性都是只读的。Page. IsPostBack 属性为 True 时表示当前网页是为响应客户端回传(PostBack，指网页及操作状态传回服务器)而加载，为 False 时表示首次加载和访问网页。

Page. IsValid 属性指示网页上的验证控件是否验证成功。若网页验证控件全部验证成功，该值为 True，否则为 False。

(3)【答】 Cookie 与 Session 类似，也是用来保存相关信息的，但 Cookie 与 Session 最大不同是，Cookie 将信息保存在客户端，而 Session 保存在服务器端。Cookie 机制采用的是在客户端保持状态的方案，而 Session 机制采用的是在服务器端保持状态的方案，由于采用服务器端保持状态的方案在客户端也需要保存一个标识，所以 Session 机制需要借助于 Cookie 机制来达到保存标识的目的。

(4)【答】 Application 对象被整个应用程序所共享，因此在使用 Application 对象存储或读取数据时，为了保证数据的一致性必须对 Application 对象进行加锁，即在同一时刻只允许一个用户对 Application 对象中的数据进行修改。引入了 Lock 和 Unlock 方法，在使用前用 Lock 方法对 Application 加锁，使用后用 Unlock 方法对其解锁，可以防止其他用户修改存储在 Application 对象中的变量，直到用户使用 Unlock 方法或超时才可再次修改。

(5)【答】 ASP .NET 网页之间常用的几种传递值的方式如下：

① 使用 URL 传值。例如，A 网页有：

```
Response.Redirect("B.aspx?a = 值 &b = 值");
```

B 网页获取传递值：

```
string a  =  Request.QueryString["a"];
string b  =  Request.QueryString["b"];
```

② 使用 Session 传值。例如，A 网页有：

```
Session["a"] = 值;
Session["b"] = 值;
Response.Redirect("B.aspx");
```

B 网页获取传递值：

```
string a = Session["a"].ToString();
string b = Session["b"].ToString();
```

③ 使用 Cookie 传值。例如，A 网页有：

```
HttpCookie mycookie = new HttpCookie("cookie");
mycookie.Value = 值;
mycookie.Expires = DateTime.Now.AddMinutes(1);          //保存 1 分钟
Response.Cookies.Add(mycookie);
Response.Redirect("B.aspx");
```

B 网页获取传递值：

```
string mystr = Request.Cookies["cookie"].Value;
```

④ 使用 Application 传值。例如，A 网页有：

```
Application["a"] = 值;
Application["b"] = 值;
Response.Redirect("B.aspx");
```

B 网页获取传递值：

```
Application.Lock();
string a = Application["a"];
string b = Application["b"];
Application.UnLock();
```

(6)【答】 Application 和 Session 对象将数据存储在服务器端。ViewState 和 Cookie 对象将数据存储在客户端。

(7)【答】 ViewState 的优缺点如下。

优点：和 Application、Session 相比耗费的服务器资源较少，视图状态数据都写入了客户端计算机中；易于维护。默认情况下，.NET 系统自动启用对控件状态数据的维护；增强的安全功能。视图状态中的值经过哈希计算和压缩，并且针对 Unicode 实现进行编码，其安全性要高于使用隐藏域。

缺点：由于视图状态存储在网页中，因此如果存储较大的值，即使在视图状态分块的情况下，用户显示网页和发送网页时的速度仍然可能减慢；视图状态存储在网页上的一个或多个隐藏域中，虽然视图状态以哈希格式存储数据，但能被篡改，如果直接查看网页输出源，能看到隐藏域中的信息，这导致潜在的安全性问题。

(8)【答】 对应的事件处理方法如下：

```
protected void Button1_Click(object sender, EventArgs e)
{    foreach (Control cnt in Page.Form.Controls)
         if (cnt is TextBox)
             (cnt as TextBox).Text = string.Empty;
}
```

第 9 章练习题

1. 单项选择题

(1) A	(2) B	(3) B	(4) C	(5) A
(6) B	(7) D	(8) A	(9) A	(10) C
(11) A	(12) D	(13) B	(14) B	(15) B
(16) D				

2. 问答题

(2)【答】 一个网页应用主题的几种方式如下：

① 在网页的页指令中指定主题。

② 在代码中指定主题。

③ 在 web.config 文件中指定主题。

④ 设置网页的 StyleSheetTheme 属性指定样式表主题。

(3)【答】 母版页的运行过程如下：

① 用户通过输入内容页的 URL 来请求某个网页。

② 获取该页之后，读取@ Page 指令。如果该指令引用一个母版页，则将读取该母版页。如果是第一次请求这两个网页，则两个网页都要进行编译。

③ 将包含更新内容的母版页合并到内容页的控件中。

④ 各个 Content 控件的内容合并到母版页中相应的 ContentPlaceHolder 控件中。最后浏览器将呈现得到后的合并页。

(4)【答】 母版页的优点如下：

① 母版页可以把网站相同的部分抽离出来，使得程序风格统一。

② 使用母版页可以集中处理网页的通用功能，以便只在一个位置上进行更新。

③ 使用母版页可以方便地创建一组控件和代码，并将结果应用于一组网页，如可以在母版页上使用控件来创建一个应用于所有页的菜单。

④ 通过控制内容占位符控件的呈现方式，母版页可以在细节上控制最终页面的布局。

(6)【答】 web.sitmap 文件是网站的一个地图文件。SiteMapDataSource 控件可以绑定到站点地图数据，并基于站点地图层次结构中指定的起始节点，在 Web 服务器控件中显示其视图。

使用 TreeView 控件进行站点导航时，可以通过与 SiteMapDataSource 控件集成实现网页的菜单功能。

第 10 章练习题

1. 单项选择题

(1) A	(2) B	(3) C	(4) C	(5) D
(6) B	(7) A	(8) C	(9) D	(10) A
(11) B	(12) B	(13) B	(14) A	(15) B
(16) A	(17) A	(18) C	(19) D	(20) A
(21) B	(22) B	(23) A	(24) A	(25) D

2. 问答题

(2)【答】 Connection对象用于数据库连接，DataSet对象作为数据存储器，DataCommand对象执行SQL命令，DataAdapter对象获取数据集合并填充DataSet对象。

(3)【答】 ① 先声明一个DataAdapter对象，然后将DataAdapter对象的SelectCommand属性设置为一个有效的Command对象。

② 创建DataAdapter对象时指定Command对象。

③ 创建DataAdapter对象时指定Select语句或者存储过程和Connection对象。

④ 创建DataAdapter对象时指定Select语句或者存储过程和连接字符串。

(4)【答】 DataReader对象和DataSet对象最大的区别是，DataReader对象使用时始终占用SqlConnection，在线操作数据库，任何对SqlConnection的操作都会引发DataReader的异常。因为DataReader每次只在内存中加载一条数据，所以占用的内存是很小的。由于DataReader的特殊性和高性能，所以DataReader对象读了第一条记录后就不能再去读取第一条记录了。

DataSet对象则将数据一次性加载在内存中，抛弃数据库连接，读取完毕即放弃数据库连接。由于DataSet将数据全部加载在内存中，所以比较消耗内存。但是确比DataReader要灵活，可以动态地添加行、列和数据，对数据库进行回传更新操作。

(5)【答】 一种是使用DataReader对象；另一种是使用DataAdapter对象。

第11章练习题

1. 单项选择题

(1) D	(2) A	(3) B	(4) C	(5) B
(6) C	(7) B	(8) D	(9) B	(10) A
(11) A	(12) B	(13) A	(14) B	(15) C
(16) D	(17) C	(18) A		

2. 问答题

(3)【答】 GridView控件提供了很多内置功能，包括可以对控件中的项进行排序、更新、删除、选择和分页。

(4)【答】 需要将AllowPaging属性设置为True，并设置PageSize属性为每页的记录个数。

(5)【答】 需要将GridView控件的DataSourceID属性设置为该数据源控件的ID值。GridView控件自动绑定到指定的数据源控件，并且可利用该数据源控件的功能来执行排序、更新、删除和分页。

(6)【答】 可以通过“字段”对话框向GridView控件添加ButtonField(按钮)字段。在单击GridView控件中的按钮时，将引发RowCommand事件，所以可以使用RowCommand事件向控件添加自定义功能。

(7)【答】 GridView控件支持删除模式，在该模式下用户可以从数据源中删除当前行。无须编写任何代码就可以将删除功能添加到GridView控件中。只需要将GridView控件的AutoGenerateDeleteButton属性设置为True便启用删除功能。

附录 B　上机实验题参考答案

第 4 章上机实验题

✍上机实验题设计

本实验题网页的源视图代码如下，从中可以看出网页的设计思路：

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title></title>
    <style type="text/css">
        <!-这里放置上机实验题所给的样式定义 ->
    </style>
  </head>
  <body>
    <form id="form1">
      <table class="auto-tabstyle">
        <tr>
          <td colspan="2" class="auto-captionstyle">我的信息</td>
        </tr>
        <tr>
          <td style="width:80px;height:25px" class="auto-tagstyle">学号</td>
          <td style="width:170px;height:25px;">
             <input id="Text1" type="text" />
          </td>
        </tr>
        <tr>
          <td class="auto-tagstyle">姓名</td>
          <td><input id="Text2" type="text" /></td>
        </tr>
        <tr>
          <td class="auto-tagstyle">性别</td>
          <td>
            <input id="Radio1" checked="true" name="sex" type="radio" value="男" />男
            <input id="Radio2" name="sex" type="radio" value="女" />女
          </td>
        </tr>
        <tr>
          <td class="auto-tagstyle">民族</td>
          <td>
            <select id="Select1" name="D1" style="width:100px;height:25px">
              <option>汉族</option>
              <option>回族</option>
              <option>满族</option>
              <option>其他</option>
            </select>
          </td>
```

```
            </tr>
            <tr>
              <td colspan = "2" style = "text - align:center;">
                 <input id = "Button1" type = "button" value = "提交" />   
                 <input id = "Reset1" type = "reset" value = "重置" />
              </td>
            </tr>
            <tr>
              <td colspan = "2" >
                <textarea id = "TextArea1" name = "S1" style = "width:100 % "></textarea>
              </td>
            </tr>
          </table>
        </form>
      </body>
    </html>
```

第 5 章上机实验题

✍上机实验题设计

本实验题网页的源视图代码如下,从中可以看出网页的设计思路:

```
<! DOCTYPE html >
< html xmlns = "http://www.w3.org/1999/xhtml">
  < head >
    < meta http - equiv = "Content - Type" content = "text/html; charset = utf - 8"/>
    < title></title>
    < script language = "javascript" type = "text/javascript">
        function disp() {
            var xh  =  document.getElementById("Text1").value;
            var xm  =  document.getElementById("Text2").value;
            var xb  =  "";
            if (document.getElementById("Radio1").checked)
                xb  =  "男";
            else
                xb  =  "女";
            var mz  =  document.getElementById("Select1").value;
            var mystr = "你的信息: ";
            mystr  +=  "学号为"  +  xh  +  ",姓名为" +  xm  +  ",性别为"  +  xb  + ",民族为"  + mz;
            document.getElementById("TextArea1").value  =  mystr;
        }
    </script >
    < style type = "text/css">
        <! - 这里放置上机实验题所给的样式定义  ->
    </style>
  </head>
  < body>
    < form id = "form1">
      < table class = "auto - tabstyle">
        < tr >
          < td colspan = "2" class = "auto - captionstyle">我的信息</td >
        </tr >
        < tr >
          < td style = "width:80px;height:25px" class = "auto - tagstyle">学号</td >
          < td style = "width:170px;height:25px;">
```

```
            <input id = "Text1" type = "text" />
          </td>
        </tr>
        <tr>
          <td class = "auto - tagstyle">姓名</td>
          <td><input id = "Text2" type = "text" /></td>
        </tr>
        <tr>
          <td class = "auto - tagstyle">性别</td>
          <td>
            <input id = "Radio1" checked = "true" name = "sex" type = "radio" value = "男" />男
            <input id = "Radio2" name = "sex" type = "radio" value = "女" />女
          </td>
        </tr>
        <tr>
          <td class = "auto - tagstyle">民族</td>
          <td>
            <select id = "Select1" name = "D1" style = "width:100px;height:25px">
              <option>汉族</option>
              <option>回族</option>
              <option>满族</option>
              <option>其他</option>
            </select>
          </td>
        </tr>
        <tr>
          <td colspan = "2" style = "text - align:center;">
           <input id = "Button1" type = "button" value = "提交" onclick = "disp()" />
             
           <input id = "Reset1" type = "reset" value = "重置" /></td>
        </tr>
        <tr>
           <td colspan = "2" >
            <textarea id = "TextArea1" name = "S1" style = "width:100 % "></textarea>
           </td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

第 6 章上机实验题

✍上机实验题设计

按照本上机实验题的设计界面设计好后，在 Button1 命令按钮上设计如下事件处理方法：

```
protected void Button_Click(object sender, EventArgs e)
{    int n;
     ArrayList myarr = new ArrayList();
     Random obj = new Random();
     Label1.Text = "";
     for (int i = 0; i < 20; i++)
     {    n = obj.Next(1, 21);
          myarr.Add(n);
          Label1.Text += n.ToString() + " ";
```

```
    }
    myarr.Sort();
    Label2.Text = "";
    foreach (int item in myarr)
        Label2.Text += item.ToString() + " ";
}
```

第 7 章上机实验题

✍上机实验题设计

本上机实验题的<body>的源视图代码如下：

```
<form id = "form1" runat = "server">
    <div>
        <span class = "auto - style1">省份: </span>
        <asp:DropDownList ID = "DropDownList1" runat = "server"
            style = "font - size: medium;height:23px;width:100px" AutoPostBack = "True"
            OnSelectedIndexChanged = "DropDownList1_SelectedIndexChanged">
        </asp:DropDownList>
        <br /><br />
        <span class = "auto - style1">城市: </span>
        <asp:DropDownList ID = "DropDownList2" runat = "server"
            style = "font - size: medium;height:23px;width:100px" >
        </asp:DropDownList>
        <br /> <br />
        <span class = "auto - style1">邮编: </span>
        <asp:TextBox ID = "TextBox1" runat = "server"
            style = "font - size: medium;height:18px;width:100px" />
        <asp:RegularExpressionValidator ID = "RegularExpressionValidator1"
           runat = "server" ControlToValidate = "TextBox1" ErrorMessage = "格式错误"
           style = "color: #800080; font - size: small; font - weight: 700;
             font - family: 仿宋" ValidationExpression = "\d{6}" />
         <asp:RequiredFieldValidator ID = "RequiredFieldValidator1"
             runat = "server" ControlToValidate = "TextBox1"
         ErrorMessage = "必须输入" style = "color: #800080; font - size: small;
              font - weight: 700; font - family: 仿宋" />
         <br /> <br />
         <asp:Button ID = "Button1" runat = "server" Text = "确定"
              style = "color: #FF0000; font - size: medium; font - weight: 700;
               font - family: 黑体" OnClick = "Button1_Click" />
         <br /> <br />
         <asp:Label ID = "Label1" runat = "server" style = "color: #FF00FF;
            font - size: medium; font - weight: 700; font - family: 仿宋" />
    </div>
</form>
```

网页上设计如下事件处理方法：

```
protected void Page_Load(object sender, EventArgs e)
{   if (!Page.IsPostBack)
    {   DropDownList1.Items.Add("");
        DropDownList1.Items.Add("湖北省");
        DropDownList1.Items.Add("江苏省");
    }
}
```

```
protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
{   DropDownList2.Items.Clear();
    if (DropDownList1.SelectedValue.ToString() == "湖北省")
    {   DropDownList2.Items.Add("武汉市");
        DropDownList2.Items.Add("黄石市");
        DropDownList2.Items.Add("荆州市");
    }
    else if (DropDownList1.SelectedValue.ToString() == "江苏省")
    {   DropDownList2.Items.Add("南京市");
        DropDownList2.Items.Add("苏州市");
        DropDownList2.Items.Add("无锡市");
    }
}
protected void Button1_Click(object sender, EventArgs e)
{   if (DropDownList1.SelectedValue.ToString() == "")
        Label1.Text = "提示：你没有选择任何省份";
    else
        Label1.Text = DropDownList1.SelectedValue
         + DropDownList2.SelectedValue + "的邮编为" + TextBox1.Text;
}
```

第 8 章上机实验题

✍上机实验题设计

在 CH8 网站的 Global.asax 文件的 Application_Start 事件处理方法中增加如下语句：

```
Application["count"] = 0;
```

网页中只有一个 Label1 控件，网页上设计如下事件处理方法：

```
protected void Page_Load(object sender, EventArgs e)
{   Application.Lock();
    Application["count"] = ((int)Application["count"]) + 1;
    Application.UnLock();
    Label1.Text = "欢迎访问本网页,你是第" + Application["count"].ToString()
     + "位访客";
}
```

第 9 章上机实验题

✍上机实验题设计

本上机实验题的<body>的源视图代码如下：

```
<form id="form1" runat="server">
    <div>
      <table class="auto-style1">
        <tr>
          <td class="auto-style2">条件设置</td>
          <td>
            <asp:Label ID="Label1" runat="server" style="color: #FF00FF;
                 font-size: medium; font-weight: 700; font-family: 仿宋" />
          </td>
        </tr>
        <tr>
```

```
            <td>
              <asp:TreeView ID = "TreeView1" runat = "server"
                OnSelectedNodeChanged = "TreeView1_SelectedNodeChanged"
                style = "color: #FF00FF; font - size: medium;
                    font - weight: 700; font - family: 仿宋">
              </asp:TreeView>
            </td>
            <td>
              <asp:ListBox ID = "ListBox1" runat = "server" Height = "142px"
                  style = "color: #0000FF; font - size: medium; font - weight: 700;
                    font - family: 楷体" Width = "231px" />
            </td>
          </tr>
        </table>
      </div>
</form>
```

在网站中新建一个 Class1.cs 类文件，在其中输入如下 student 类代码：

```
public class student
{   public int xh { set; get; }               //学号
    public string xm { set; get; }            //姓名
    public string xb { set; get; }            //性别
    public string mz { set; get; }            //民族
    public string bh { set; get; }            //班号
    public student(int xh1, string xm1, string xb1, string mz1, string bh1)
    {   xh = xh1; xm = xm1; xb = xb1;
        mz = mz1; bh = bh1;
    }
    public string getstudent()
    {   string mystr;
        mystr = xh.ToString() + "----" + xm + "--"
          + xb + "--" + mz + "--" + bh;
        return mystr;
    }
}
```

在本网页上设计如下事件处理方法：

```
protected void Page_Load(object sender, EventArgs e)
{   if (!Page.IsPostBack)
    {   TreeView1.Nodes.Clear();
        TreeNode node = new TreeNode("选择");
        TreeView1.Nodes.Add(node);
        node = new TreeNode("性别");
        TreeView1.Nodes[0].ChildNodes.Add(node);
        node = new TreeNode("男");
        TreeView1.Nodes[0].ChildNodes[0].ChildNodes.Add(node);
        node = new TreeNode("女");
        TreeView1.Nodes[0].ChildNodes[0].ChildNodes.Add(node);
        node = new TreeNode("班号");
        TreeView1.Nodes[0].ChildNodes.Add(node);
        node = new TreeNode("15001");
        TreeView1.Nodes[0].ChildNodes[1].ChildNodes.Add(node);
        node = new TreeNode("15002");
        TreeView1.Nodes[0].ChildNodes[1].ChildNodes.Add(node);
```

```
        }
    }
    protected void TreeView1_SelectedNodeChanged(object sender, EventArgs e)
    {   int depth = TreeView1.SelectedNode.Depth;
        if (depth == 2)
        {   student[] st = new student[5] { new student(1,"王华","女","汉族","15001"),
                new student(2,"孙丽","女","满族","15002"),
                new student(3,"李兵","男","汉族","15001"),
                new student(6,"张军","男","汉族","15001"),
                new student(8,"马棋","男","回族","15002") };
            string fieldname = "";                //字段名
            fieldname = TreeView1.SelectedNode.Parent.Text.ToString();
            string fieldvalue;                    //字段值
            fieldvalue = TreeView1.SelectedNode.Text.ToString();
            ListBox1.Items.Clear();
            Label1.Text = "条件:" + fieldname + "=" + fieldvalue;
            ListBox1.Items.Add("学号-姓名-性别-民族--班号");
            foreach (student s in st)
            {   if (fieldname == "性别")
                {   if (s.xb == fieldvalue)
                        ListBox1.Items.Add(s.getstudent());
                }
                else if (fieldname == "班号")
                {   if (s.bh == fieldvalue)
                        ListBox1.Items.Add(s.getstudent());
                }
            }
        }
        else
        {   Label1.Text = "条件设置不正确";
            ListBox1.Items.Clear();
        }
    }
```

第 10 章上机实验题

✍上机实验题设计

按照本上机实验题的设计界面设计好后，在 Button1 和 Button2 按钮上设计如下事件处理方法：

```
    protected void Button1_Click(object sender, EventArgs e)
    {   string mystr, mysql;
        SqlConnection myconn = new SqlConnection();
        SqlCommand mycmd = new SqlCommand();
        mystr = System.Configuration.ConfigurationManager.
            ConnectionStrings["myconnstring"].ToString();
        myconn.ConnectionString = mystr;
        myconn.Open();
        mysql = "SELECT student.学号,student.姓名,AVG(score.分数) AS 平均分 ";
        mysql += "FROM student,score ";
        mysql += "WHERE student.学号 = score.学号 ";
        mysql += "GROUP BY student.学号,student.姓名";
        mycmd.CommandText = mysql;
        mycmd.Connection = myconn;
```

```
    SqlDataReader myreader = mycmd.ExecuteReader();
    ListBox1.Items.Clear();
    ListBox1.Items.Add("学号 ------- 姓名 -------- 平均分");
    ListBox1.Items.Add(" ============================== ");
    while (myreader.Read())                //循环读取信息
        ListBox1.Items.Add(String.Format("{0} ---------- {1} -------- {2}",
            myreader["学号"].ToString(), myreader[1].ToString(),
            myreader.GetDouble(2)));
    myconn.Close();
    myreader.Close();
}
protected void Button2_Click(object sender, EventArgs e)
{   string mystr, mysql;
    SqlConnection myconn = new SqlConnection();
    mystr = System.Configuration.ConfigurationManager.
        ConnectionStrings["myconnstring"].ToString();
    myconn.ConnectionString = mystr;
    myconn.Open();
    mysql = "SELECT course.课程号,course.课程名,AVG(score.分数) AS 平均分 ";
    mysql += "FROM course,score ";
    mysql += "WHERE course.课程号 = score.课程号 ";
    mysql += "GROUP BY course.课程号,course.课程名";
    SqlDataAdapter myda = new SqlDataAdapter(mysql, myconn);
    myconn.Close();
    DataSet mydataset = new DataSet();
    myda.Fill(mydataset, "mydata");
    ListBox1.Items.Clear();
    ListBox1.Items.Add("课程号 ----- 课程名 ------ 平均分");
    ListBox1.Items.Add(" ============================== ");
    foreach (DataRow dr in mydataset.Tables[0].Rows)
        ListBox1.Items.Add(String.Format("{0} ------- {1} ------ {2}",
            dr[0].ToString(), dr[1].ToString(), dr[2].ToString()));
}
```

第 11 章上机实验题

✍上机实验题设计

本上机实验题的<body>的源视图代码如下：

```
<form id = "form1" runat = "server">
  <div>
     <span class = "auto-style1">班号: </span>
     <asp:DropDownList ID = "DropDownList1" runat = "server"
        DataSourceID = "SqlDataSource2" DataTextField = "班号"
        DataValueField = "班号" Height = "18px" Width = "104px"
        style = "font-size: medium; font-weight: 700; font-family: 仿宋">
     </asp:DropDownList>
     <asp:SqlDataSource ID = "SqlDataSource2" runat = "server"
         ConnectionString = "<%$ ConnectionStrings:schoolConnectionString %>"
         SelectCommand = "SELECT distinct 班号 FROM student">
     </asp:SqlDataSource>
     <asp:Button ID = "Button1" runat = "server" Text = "确定"
       style = "color: #FF0000; font-size: medium; font-weight: 700;
         font-family: 黑体" />
     <br /><br />
```

```
        <span class = "auto - style2">查询结果：</span>
        <br />
    </div>
    <asp:GridView ID = "GridView1" runat = "server" AutoGenerateColumns = "False"
      DataKeyNames = "学号" DataSourceID = "SqlDataSource1"
      BackColor = "LightGoldenrodYellow" BorderColor = "Tan" BorderWidth = "1px"
      CellPadding = "2" GridLines = "None" style = "font - weight: 700; font - family: 仿宋"
      ForeColor = "Black">
      <AlternatingRowStyle BackColor = "PaleGoldenrod" />
        <Columns>
          <asp:BoundField DataField = "学号" HeaderText = "学号" ReadOnly = "True"
              SortExpression = "学号" />
          <asp:BoundField DataField = "姓名" HeaderText = "姓名" SortExpression = "姓名" />
          <asp:BoundField DataField = "平均分" HeaderText = "平均分" ReadOnly = "True"
            SortExpression = "平均分" />
        </Columns>
        <FooterStyle BackColor = "Tan" />
        <HeaderStyle BackColor = "Tan" Font - Bold = "True" />
        <PagerStyle BackColor = "PaleGoldenrod" ForeColor = "DarkSlateBlue"
          HorizontalAlign = "Center" />
        <SelectedRowStyle BackColor = "DarkSlateBlue" ForeColor = "GhostWhite" />
        <SortedAscendingCellStyle BackColor = "#FAFAE7" />
        <SortedAscendingHeaderStyle BackColor = "#DAC09E" />
        <SortedDescendingCellStyle BackColor = "#E1DB9C" />
        <SortedDescendingHeaderStyle BackColor = "#C2A47B" />
      </asp:GridView>
      <asp:SqlDataSource ID = "SqlDataSource1" runat = "server"
        ConnectionString = "<%$ ConnectionStrings:schoolConnectionString %>"
        SelectCommand = "SELECT student.学号,student.姓名,AVG(score.分数) AS 平均分
          FROM student,course,score
          WHERE student.学号 = score.学号 AND course.课程号 = score.课程号
            AND student.班号 = @bh
          GROUP BY student.学号,student.姓名
          ORDER BY AVG(score.分数) DESC">
        <SelectParameters>
          <asp:ControlParameter ControlID = "DropDownList1" Name = "bh"
            PropertyName = "SelectedValue" />
        </SelectParameters>
    </asp:SqlDataSource>
</form>
```

网页中不设计任何事件处理方法，其中“确定”按钮仅仅起到提交网页的作用。

参 考 文 献

[1] Jason N Gaylord,Christian Wenz,Pranav Rastogi,Todd Miranda,Scott Hanselman. ASP .NET 4.5 高级编程[M]. 8 版. 李增民,苗荣译. 北京：清华大学出版社,2014.

[2] Imar Spaanjaars. ASP .NET 4.5 入门经典[M]. 7 版. 刘楠,陈晓宇译. 北京：清华大学出版社,2013.

[3] Mary Delamater,Anne Boehm. Murach's ASP .NET 4.5 Web Programming with C# 2012[M]. Mumbai：Mike Murach & Associates,Inc. 2013.

[4] Adam Freeman,Matthew MacDonald,Mario Szpuszta. Pro ASP .NET 4.5 in C#[M]. New York：Appess,2013.

[5] Matthew MacDonald,Adam Freeman,Mario Szpuszta. ASP .NET 4 高级程序设计[M]. 4 版. 博思工作室译. 北京：人民邮电出版社,2011.

[6] 郑阿奇. ASP .NET 4.0 实用教程[M]. 北京：电子工业出版社,2013.

[7] Mitchell S. ASP .NET 2.0 入门经典[M]. 陈武译. 北京：人民邮电出版社,2007.

[8] Liberty J. 学习 ASP .NET 2.0 和 AJAX[M]. 刘平利译. 北京：机械工业出版社,2008.

[9] Onion F. Essential ASP .NET 中文版[M]. 袁国忠译. 北京：人民邮电出版社,2007.

[10] 赵晓东,张正礼,许小荣. ASP .NET 3.5 从入门到精通[M]. 北京：清华大学出版社,2009.

[11] 闫洪亮,潘勇. ASP .NET 程序设计教程[M]. 上海：上海交通大学出版社,2006.

[12] 张跃廷. ASP .NET 2.0 自学手册[M]. 北京：人民邮电出版社,2008.

[13] 马骏. ASP .NET 网页设计与网站开发[M]. 北京：人民邮电出版社,2007.

[14] 王院峰. 零基础学 ASP .NET 2.0[M]. 北京：机械工业出版社,2008.

[15] 贺伟,陈哲,龚涛,戴博. 新一代 ASP .NET 2.0 网络编程入门与实践[M]. 北京：清华大学出版社,2007.

[16] 李春葆. ASP .NET 动态网站设计教程——基于 C#＋SQL Server[M]. 北京：清华大学出版社,2011.

[17] 李春葆. ASP .NET 2.0 动态网站设计教程——基于 VB＋Access[M]. 北京：清华大学出版社,2010.

[18] 李春葆. ASP .NET 2.0 动态网站设计教程——基于 C#＋Access[M]. 北京：清华大学出版社,2010.

[19] 李春葆. ASP 动态网页设计——基于 SQL Server 2005 数据库[M]. 北京：清华大学出版社,2009.

[20] 李春葆. C#程序设计教程[M]. 3 版. 北京：清华大学出版社,2015.